Can Languages and Tools Produce Better Software?

- In this talk, better means more reliable
  - Less likely to fail

SPT Approach

- Redundency is good
- Redundancy exposes inconsistency

SPT Approach

- Redundency is good
- Redundancy exposes inconsistency
- Inconsistency points to errors

SPT Approach

- Redundency is good
- Redundancy exposes inconsistency
- Inconsistency points to errors
- Compare what programmer should do against what code does

Lightweight Specifications

- Start with description of correct/incorrect behavior
  - Not total specification $\Rightarrow$ not total correctness
- Programmers provide specs
- Tools reward additional effort
  - Mechanically and systematically find errors

Programmers Write Types

- Successful, wide accepted specifications
  - Programmers understand and write type declarations
  - Declarations aid comprehension
  - Type checking is fast, routine, and finds errors
- Serve several roles
  - Documentation of interface syntax
  - Basis for program abstractions
  - Static (compile-time) error detection

Extend Successful Approach

- Specify and check other program properties
  - Languages (programming and domain-specific) express properties
  - Tools check that code obeys rules
- Goal is partial correctness
  - Detect and report important classes of errors
  - No guarantee of program correctness
- Systematic
  - Sound, complete analysis can flag absence of errors

SPT Overview

- Reliable interfaces
  - Vault
  - SLAM
  - ESP
- Concurrency and security
  - Behave!
  - Golf

Interfaces Have Usage Rules

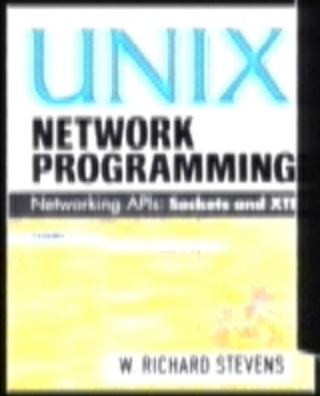
- Specs in documentation
  - Incomplete, unenforced, wordy
  - Order of operations & data access
  - Resource management
- Disobeying rules causes bad behavior
  - System crash or deadlock
  - Unexpected exceptions
  - Failed runtime checks
  - Security flaws

Example Usage Rule: Sockets

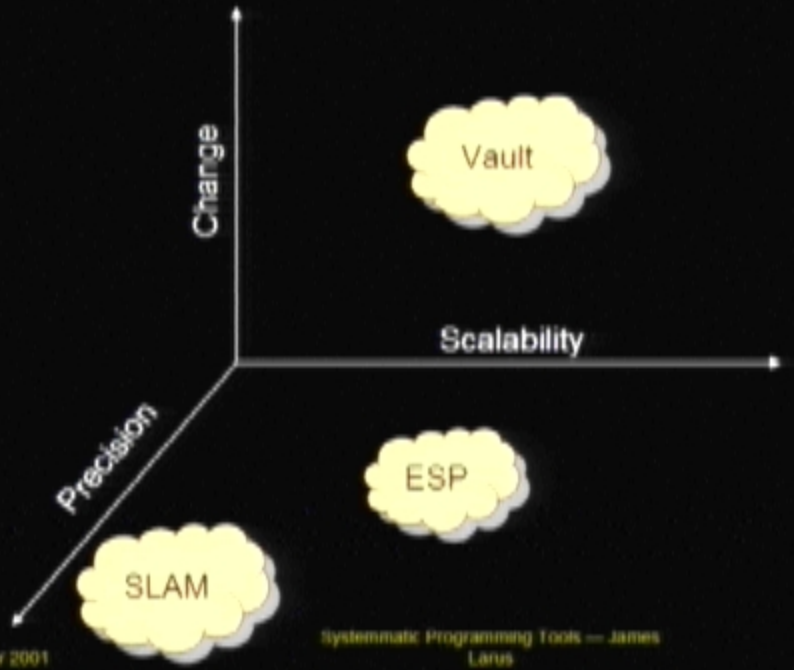
the "communication domain" in which communication is to take place; see protocols(5).

Sockets of type SOCK_STREAM are full-duplex byte streams, similar to pipes. A stream socket must be in a connected state before any data may be sent or received on it. A connection to another socket is created with a connect(2) call. Once connected, data may be transferred using read(2V) and write(2V) calls or some variant of the send(2) and recv(2) calls. When a session has been completed a close(2V), may be performed. Out-of-band data may also be transmitted as described in send(2) and received as described in recv(2).

The communications protocols used to implement a SOCK_STREAM insure that data is not lost or duplicated. If a piece of



SPT Specification & Checking



(not to scale)

SPT Overview

- Reliable interfaces
  - Vault
  - SLAM
  - ESP
- Security and concurrency
  - Behave!
  - Golf

Vault (Rob DeLine, Manuel Fahndrich)

- Vault folds usage rules into programming language's type system
  - Interface author states rules in type signature
  - Compiler rejects client code that violates rule (type error)
  - Every violation guaranteed to be found (soundness)

Why a Type-based Approach?

- Types are widely accepted form of specification
- Specification and code are kept in sync
- Developers find and fix bugs before execution
- Programmers understand violations and how to fix them
- Efficient, scalable analysis doesn't slow development

Tracked Types and Keys

- Usage rules talk about individual objects

```
void listen (tracked(S) sock, int)    [ S @ named -> listening ]
```

Tracked Types and Keys

- Usage rules talk about individual objects

```
void listen (tracked(S) sock, int)    [ S @ named -> listening ]
```

Tracked Types and Keys

- Usage rules talk about individual objects

```
void listen (tracked(S) sock, int)    [ S @ named -> listening ]
```

- Naming: tracked types give compile-time names (keys) to objects

Tracked Types and Keys

- Usage rules talk about individual objects

```
void listen (tracked(S) sock, int)    [ S @ named -> listening ]
```

- Naming: tracked types give compile-time names (keys) to objects
- Object state: key can have symbolic state to model state of object

Tracked Types and Keys

- Change specs combine pre/post condition for functions

Tracked Types and Keys

- Change specs combine pre/post condition for functions
- Add key:
  - `tracked(S) sock socket (...) [ new S @ raw ]`
- Consume key:
  - `void close (tracked(S) sock) [ -S ]`
- Change key state:
  - `void bind (tracked(S) sock) [ S @ raw -> named]`

Example: Sockets Interface

```
tracked(S) sock socket(domain, style, int)          [new S @ raw]

void           bind(tracked(S) sock, sockaddr)      [S @ raw -> named]

void           listen(tracked(S) sock, int)         [S @ named -> listening]

tracked(N) sock accept(tracked(S) sock)             [S @ listening,
                                                     new N @ ready]

void           receive(tracked(S) sock, byte[])     [S @ ready]

void           close(tracked(S) sock)               [-S]
```

Example: Sockets Interface

```
tracked(S) sock socket(domain, style, int)          [new S @ raw]

void           bind(tracked(S) sock, sockaddr)      [S @ raw -> named]

void           listen(tracked(S) sock, int)         [S @ named -> listening]

tracked(N) sock accept(tracked(S) sock)             [S @ listening,
                                                     new N @ ready]

void           receive(tracked(S) sock, byte[])     [S @ ready]

void           close(tracked(S) sock)               [-S]
```

Statically Enforcing Usage Protocols

- At every program point, Vault compiler computes key set
  - Functions and built-in operations (new/free) change key set
  - Key must be consumed, not "leaked"
- On each path in function's body, check:
  - Pre-condition transformed into post-condition
  - All proof obligations satisfied
    - Pre-conditions of other function calls
    - Primitive operations (memory access, free)
- Avoid exponential blow-up
  - Require uniform predicate at join points

Checking Socket Client Code

```
void communicate(sockaddr addr) {
  tracked(K) sock s = socket(`UNIX, `INET, 0);
  bind(s, addr);
  listen(s, 0);
  while (!shutdown()) {
    tracked(J) sock c = accept(s);
    receive(c, buffer);
    close(c);
  }
  close(s);
}
```

Checking Socket Client Code

```
void communicate(sockaddr addr) {                    { }
  tracked(K) sock s = socket(`UNIX, `INET,0);        { K@raw }
  bind(s,addr);                                      { K@named }
  listen(s, 0);                                      { K@listening }
  while (!shutdown()) {                              { K@listening }
    tracked(J) sock c = accept(s);                   { K@listening, J@ready }
    receive(c, buffer);                              { K@listening, J@ready }
    close(c);                                        { K@listening }
  }                                                  { K@listening }
  close(s);                                          { }
}
```

Checking Socket Client Code

```
void communicate(sockaddr addr) {
  tracked(K) sock s = socket(`UNIX, `INET,0);
  bind(s, addr);
  listen(s, 0);
  while (!shutdown()) {
    tracked(J) sock c = accept(s);
    receive(c, buffer);
    close(c);
  }
  close(s);
}
```

Checking Socket Client Code

```
void communicate(sockaddr addr) {
  tracked(K) sock s = socket(`UNIX, `INET,0);
  bind(s, addr);
  listen(s, 0);
  while (!shutdown()) {
    tracked(J) sock c = accept(s);
    receive(c, buffer);
    close(c);
  }
  close(s);
}
```

Checking (Wrong) Socket Client Code

```
void communicate(sockaddr addr) {
  tracked(K) sock s = socket(`UNIX, `INET,0);
  bind(s, addr);
  listen(s, 0);
  while (!shutdown()) {
    tracked(J) sock c = accept(s);
    receive(c, buffer);
    close(s);
  }
  close(s);
}
```

Checking (Wrong) Socket Client Code

```
void communicate(sockaddr addr) {          { }
  tracked(K) sock s = socket(`UNIX, `INET,0);      { K@raw }
  bind(s, addr);                           { K@named }
  listen(s, 0);                            { K@listening }
  while (!shutdown()) {                    { K@listening }
    tracked(J) sock c = accept(s);         { K@listening, J@ready }
    receive(c, buffer);                    { K@listening, J@ready }
    close(s);                              { J@ready }
  }                                        { }
  close(s);                                { }
}
```

Checking (Wrong) Socket Client Code

```
void communicate(sockaddr addr) {          { }
  tracked(K) sock s = socket(`UNIX, `INET,0);      { K@raw }
  bind(s, addr);                           { K@named }
  listen(s, 0);                            { K@listening }
  while (!shutdown()) {                    { K@listening }
    tracked(J) sock c = accept(s);         { K@listening, J@ready }
    receive(c, buffer);                    { K@listening, J@ready }
    close(s);                              { J@ready }
  }                                        { }
  close(s);                                { }
}
```

No agreement at join point

Checking (Wrong) Socket Client Code

```
void communicate(sockaddr addr) {          { }
  tracked(K) sock s = socket(`UNIX, `INET,0);      { K@raw }
  bind(s ,addr);                           { K@named }
  listen(s, 0);                            { K@listening }
  while (!shutdown()) {                    { K@listening }
    tracked(J) sock c = accept(s);         { K@listening, J@ready }
    receive(c, buffer);                    { K@listening, J@ready }
    close(s);                              { J@ready }
  }                                        { }
  close(s);                                { }
}
```

✘ Leaking key T

Checking (Wrong) Socket Client Code

```
void communicate(sockaddr addr) {          { }
  tracked(K) sock s = socket(`UNIX, `INET,0);      { K@raw }
  bind(s, addr);                           { K@named }
  listen(s,0);                             { K@listening }
  while (!shutdown()) {                    { K@listening }
    tracked(J) sock c = accept(s);         { K@listening, J@ready }
    receive(c ,buffer);                    { K@listening, J@ready }
    close(s);                              { J@ready }
  }                                        { }
  close(s);                                { }
}
```

✘ Botched precondition

Vault Research

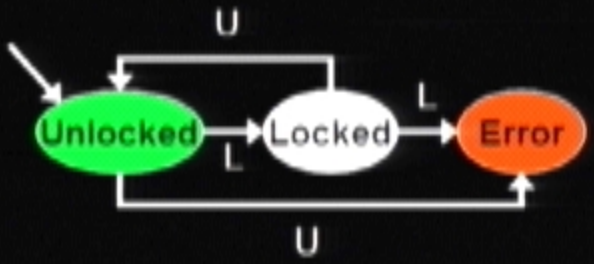
- Few new language features express and check many usage rules
  - Still imperative programming style
  - Handle “real” applications: device drivers, DirectX
  - Check “real” rules that developers commonly violate
- On-going research
  - Usage rules in C#
  - Safe, explicit memory management in CLR

SLAM (Tom Ball, Sriram Rajamani)

- Input
  - C source code, "as is"
  - API rules in SLIC language
- Automatically create abstraction of C program
  - Abstract model = Boolean program
- Systematic exploration of model's state space
  - Does feasible path through program lead to error state in SLIC?
- Demand-driven refinement of model
  - Exclude infeasible paths

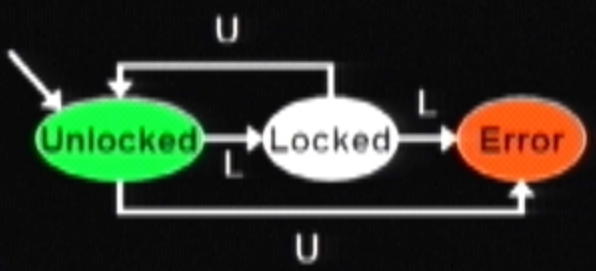
API Rule for Locking

State Machine



API Rule for Locking

State Machine



SLIC

```
state {
   int locked = 0;
}

Lock.call {
  if (locked==1) abort;
  else locked = 1;
}

UnLock.call {
  if (locked==0) abort;
  else locked = 0;
}
```

Is Locking Protocol Respected?

```
do {
     //get the write lock
    KeAcquireSpinLock(&devExt->writeListLock);
    nPacketsOld = nPackets;
    request = devExt->WriteListHeadVa;

    if (request && request->status){
          devExt->WriteListHeadVa = request->Next;
          KeReleaseSpinLock(&devExt->writeListLock);
          irp = request->irp;
          if (request->status > 0){
                    irp->IoStatus.Status = STATUS_SUCCESS;
                    irp->IoStatus.Information = request->Status;
                    }
          else {
                    irp->IoStatus.Status = STATUS_UNSUCCESSFUL;
                    irp->IoStatus.Information = request->Status;
          }
          SmartDevFreeBlock(request);
          IoCompleteRequest(irp, IO_NO_INCREMENT);
          nPackets++;
    }
} while (nPackets != nPacketsOld);

KeReleaseSpinLock(&devExt->writeListLock);
```

SLIC Instrumented Code

```
do {
     //get the write lock
    Lock.call();
    KeAcquireSpinLock(&devExt->writeListLock);
    nPacketsOld = nPackets;
    request = devExt->WriteListHeadVa;

    if (request && request->status){
          devExt->WriteListHeadVa = request->Next;
          UnLock.call();
          KeReleaseSpinLock(&devExt->writeListLock);
          irp = request->irp;
          if (request->status > 0){
                    ...
                   }
          else {
                    ...
          }
          SmartDevFreeBlock(request);
          IoCompleteRequest(irp, IO_NO_INCREMENT);
          nPackets++;
    }
} while (nPackets != nPacketsOld);
UnLock.call();
KeReleaseSpinLock(&devExt->writeListLock);
```

State-based Search Fails

```
do {
     //get the write lock
   Lock.call();
   KeAcquireSpinLock(&devExt->writeListLock);

   nPacketsOld = nPackets;
   request = devExt->WriteListHeadVa;

   if (request && request->status){
        devExt->WriteListHeadVa = request->Next;
        UnLock.call();
        KeReleaseSpinLock(&devExt->writeListLock);
        ...
        nPackets++;
   }
} while (nPackets != nPacketsOld);
UnLock.call();
KeReleaseSpinLock(&devExt->writeListLock);
```

State-based Search Fails

```
do {
     //get the write lock
   Lock.call();
   KeAcquireSpinLock(&devExt->writeListLock);

   nPacketsOld = nPackets;
   request = devExt->WriteListHeadVa;

   if (request && request->status){
        devExt->WriteListHeadVa = request->Next;
        UnLock.call();
        KeReleaseSpinLock(&devExt->writeListLock);
        ...
        nPackets++;
   }
} while (nPackets != nPacketsOld);
UnLock.call();
KeReleaseSpinLock(&devExt->writeListLock);
```

State-based Search Fails

```
do {
     //get the write lock
   Lock.call();
   KeAcquireSpinLock(&devExt->writeListLock);

   nPacketsOld = nPackets;
   request = devExt->WriteListHeadVa;

   if (request && request->status){
        devExt->WriteListHeadVa = request->Next;
        UnLock.call();
        KeReleaseSpinLock(&devExt->writeListLock);
        ...
        nPackets++;
   }
} while (nPackets != nPacketsOld);
UnLock.call();
KeReleaseSpinLock(&devExt->writeListLock);
```

State-based Search Fails

```
do {
     //get the write lock
   Lock.call();
   KeAcquireSpinLock(&devExt->writeListLock);

   nPacketsOld = nPackets;
   request = devExt->WriteListHeadVa;

   if (request && request->status){
        devExt->WriteListHeadVa = request->Next;
        UnLock.call();
        KeReleaseSpinLock(&devExt->writeListLock);
        ...
        nPackets++;
   }
} while (nPackets != nPacketsOld);
UnLock.call();
KeReleaseSpinLock(&devExt->writeListLock);
```

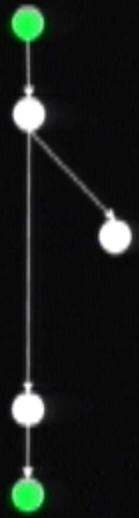
State-based Search Fails

```
do {
     //get the write lock
   Lock.call();
   KeAcquireSpinLock(&devExt->writeListLock);

   nPacketsOld = nPackets;
   request = devExt->WriteListHeadVa;

   if (request && request->status){
        devExt->WriteListHeadVa = request->Next;
        UnLock.call();
        KeReleaseSpinLock(&devExt->writeListLock);
        ...
        nPackets++;
   }
} while (nPackets != nPacketsOld);
UnLock.call();
KeReleaseSpinLock(&devExt->writeListLock);
```

State-based Search Fails

```
do {
     //get the write lock
   Lock.call();
   KeAcquireSpinLock(&devExt->writeListLock);

   nPacketsOld = nPackets;
   request = devExt->WriteListHeadVa;

   if (request && request->status){
        devExt->WriteListHeadVa = request->Next;
        UnLock.call();
        KeReleaseSpinLock(&devExt->writeListLock);
        ...
        nPackets++;
   }
} while (nPackets != nPacketsOld);
UnLock.call();
KeReleaseSpinLock(&devExt->writeListLock);
```

State-based Search Fails



```
do {
     //get the write lock
   Lock.call();
   KeAcquireSpinLock(&devExt->writeListLock);

   nPacketsOld = nPackets;
   request = devExt->WriteListHeadVa;

   if (request && request->status){
        devExt->WriteListHeadVa = request->Next;
        UnLock.call();
        KeReleaseSpinLock(&devExt->writeListLock);
        ...
        nPackets++;
   }
} while (nPackets != nPacketsOld);
UnLock.call();
KeReleaseSpinLock(&devExt->writeListLock);
```

State-based Search Fails

```
do {
     //get the write lock
   Lock.call();
   KeAcquireSpinLock(&devExt->writeListLock);

   nPacketsOld = nPackets;
   request = devExt->WriteListHeadVa;

   if (request && request->status){
        devExt->WriteListHeadVa = request->Next;
        UnLock.call();
        KeReleaseSpinLock(&devExt->writeListLock);
        ...
        nPackets++;
   }
} while (nPackets != nPacketsOld);
UnLock.call();
KeReleaseSpinLock(&devExt->writeListLock);
```

State-based Search Fails

```
do {
     //get the write lock
   Lock.call();
   KeAcquireSpinLock(&devExt->writeListLock);

   nPacketsOld = nPackets;
   request = devExt->WriteListHeadVa;

   if (request && request->status){
        devExt->WriteListHeadVa = request->Next;
        UnLock.call();
        KeReleaseSpinLock(&devExt->writeListLock);
        ...
        nPackets++;
   }
} while (nPackets != nPacketsOld);
UnLock.call();
KeReleaseSpinLock(&devExt->writeListLock);
```

State-based Search Fails

```
do {
      //get the write lock
    Lock.call();
    KeAcquireSpinLock(&devExt->writeListLock);

    nPacketsOld = nPackets;
    request = devExt->WriteListHeadVa;

    if (request && request->status){
         devExt->WriteListHeadVa = request->Next;
         UnLock.call();
         KeReleaseSpinLock(&devExt->writeListLock);
         ...
         nPackets++;
    }
} while (nPackets != nPacketsOld);
UnLock.call();
KeReleaseSpinLock(&devExt->writeListLock);
```

Boolean Program; Abstraction 1

```
do
    //get the write lock
   Lock.call();

   if (*) then
       UnLock.call();

       if (*) then

       else

       fi

   fi
while (*);
UnLock.call();
```

Boolean Program; Abstraction 1

```
do
    //get the write lock
   Lock.call();

   if (*) then
       UnLock.call();

       if (*) then

       else

       fi

   fi
while (*);
UnLock.call();
```

Boolean Program; Abstraction 1

```
do
    //get the write lock
   Lock.call();

   if (*) then
       UnLock.call();

       if (*) then

       else

       fi

   fi
while (*);
UnLock.call();
```

Boolean Program; Abstraction 1

```
do
    //get the write lock
   Lock.call();

   if (*) then
       UnLock.call();

       if (*) then

       else

       fi

   fi
while (*);
UnLock.call();
```

Boolean Program; Refine Abstraction

- Examine infeasible path
- Introduce boolean variable b to represent:
  nPacketsOld == nPackets

```
do
    //get the write lock
   Lock.call();
   npacketsOld = npackets;
   if (*) then
        UnLock.call();
        if (*) then
        else
        fi
        npackets++;
   fi
while(nPackets!=nPacketsOld)
UnLock.call();
```

Boolean Program; Refine Abstraction

- Examine infeasible path
- Introduce boolean variable b to represent:
  nPacketsOld == nPackets

```
do
    //get the write lock
   Lock.call();
   npacketsOld = npackets;
   if (*) then
        UnLock.call();
        if (*) then
        else
        fi
        npackets++;
   fi
while(nPackets!=nPacketsOld)
UnLock.call();
```

Boolean Program; Refine Abstraction

- Examine infeasible path
- Introduce boolean variable b to represent:
  nPacketsOld == nPackets

```
do
    //get the write lock
   Lock.call();
   b = true;
   if (*) then
       UnLock.call();
       if (*) then
       else
       fi
       b := b ? false : *;
   fi
while (!b)
UnLock.call();
```

Boolean Program; Abstraction 2

```
do
    //get the write lock
   Lock.call();

   b := true;

   if (*) then
        UnLock.call();

        if (*) then

        else

        fi

        b := b ? false : *;
   fi

while ( !b );

UnLock.call();
```

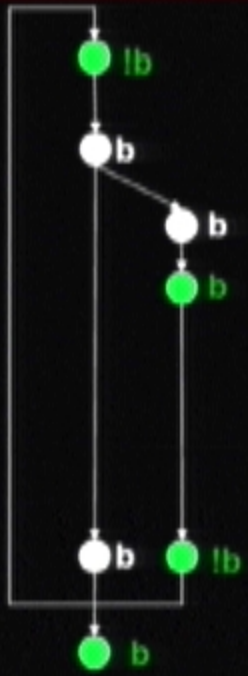
Boolean Program; Abstraction 2


```
do
    //get the write lock
    Lock.call();
    b := true;
    if (*) then
        UnLock.call();
        if (*) then
        else
        fi
        b := b ? false : *;
    fi
while ( !b );
UnLock.call();
```

Boolean Program; Abstraction 2

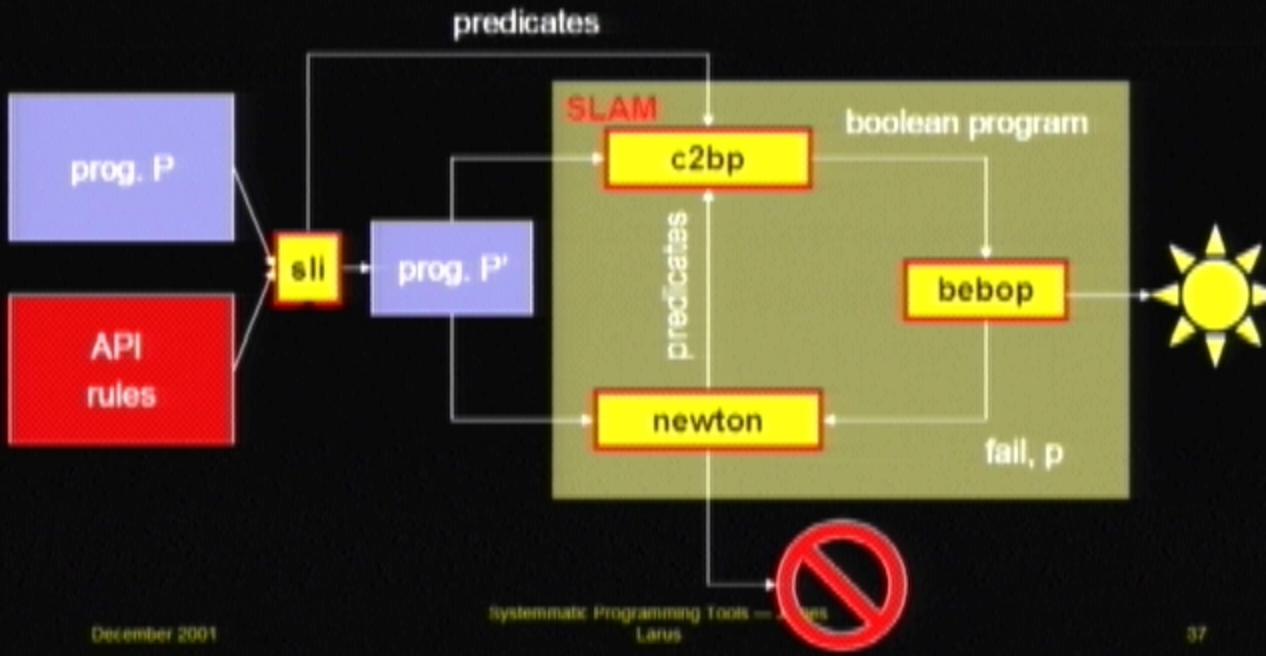

```
do
   //get the write lock
   Lock.call();
   b := true;
   if (*) then
      UnLock.call();
      if (*) then
      else
      fi
      b := b ? false : *;
   fi
while ( !b );
UnLock.call();
```

Boolean Program; Abstraction 2




```
do
    //get the write lock
    Lock.call();
    b := true;
    if (*) then
        UnLock.call();
        if (*) then
        else
        fi
        b := b ? false : *;
    fi
while ( !b );
UnLock.call();
```

Checking API Rules with SLAM



SLAM Summary

- Automatic analysis (no code annotation)
  - SLIC specification of API rules
- Report errors with source-level paths
- Very precise analysis
  - Minimize false positives
- Current research
  - Applying to device drivers
  - Tool scalability

The Static Driver Verifier

Thomas Ball
Sriram K. Rajamani

Software Productivity Tools

Programmer Productivity
Research Center





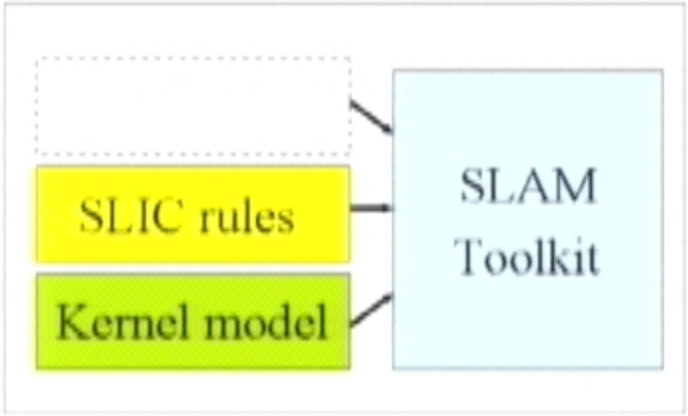

The Static Driver Verifier

Thomas Ball
Sriram K. Rajamani

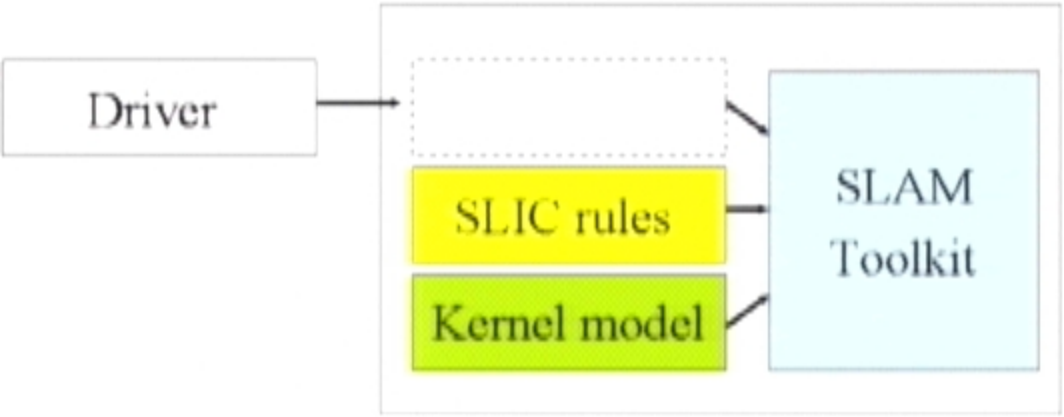
Software Productivity Tools

Programmer Productivity
Research Center

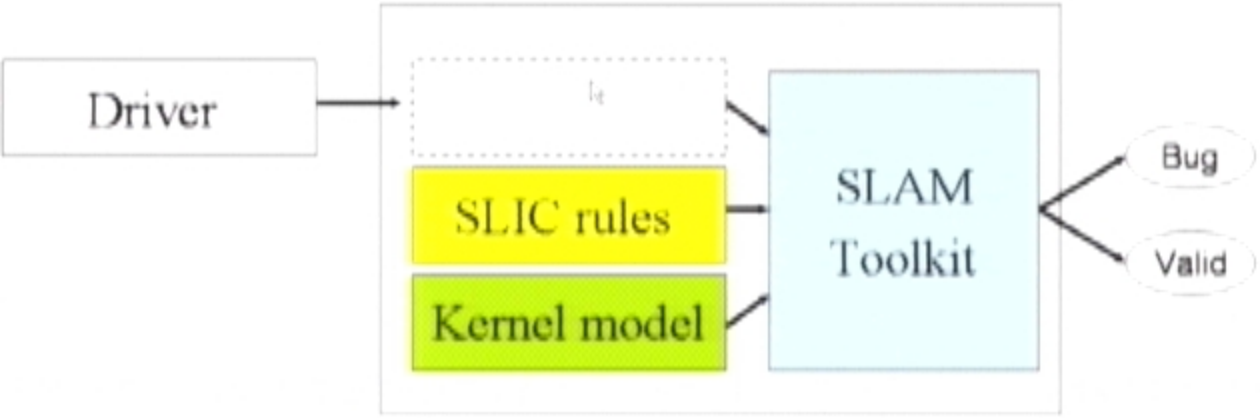
Static Driver Verifier



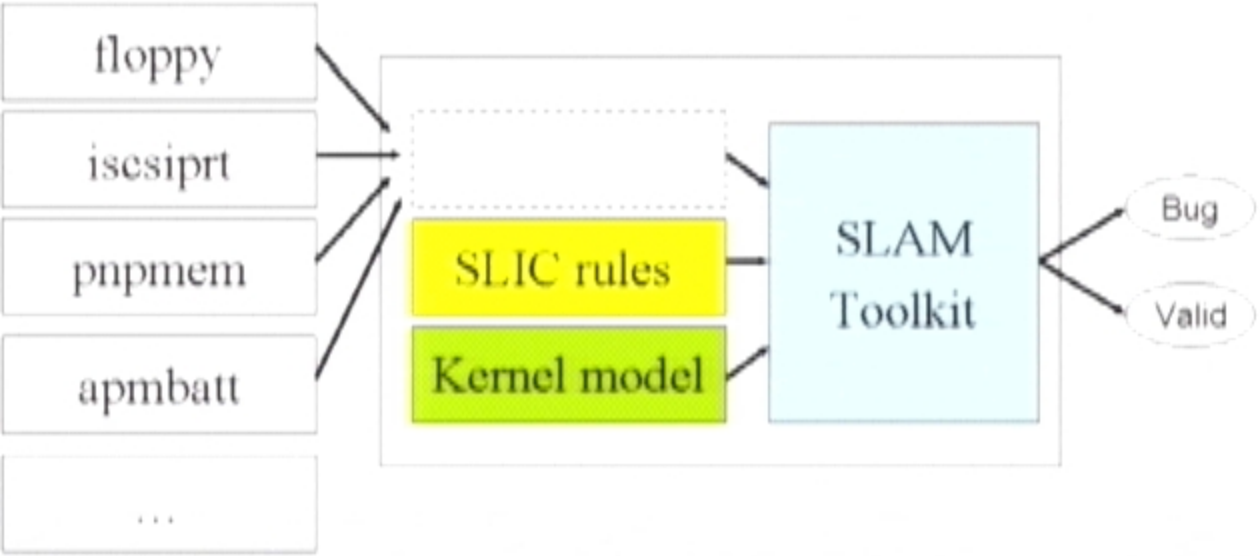
Static Driver Verifier

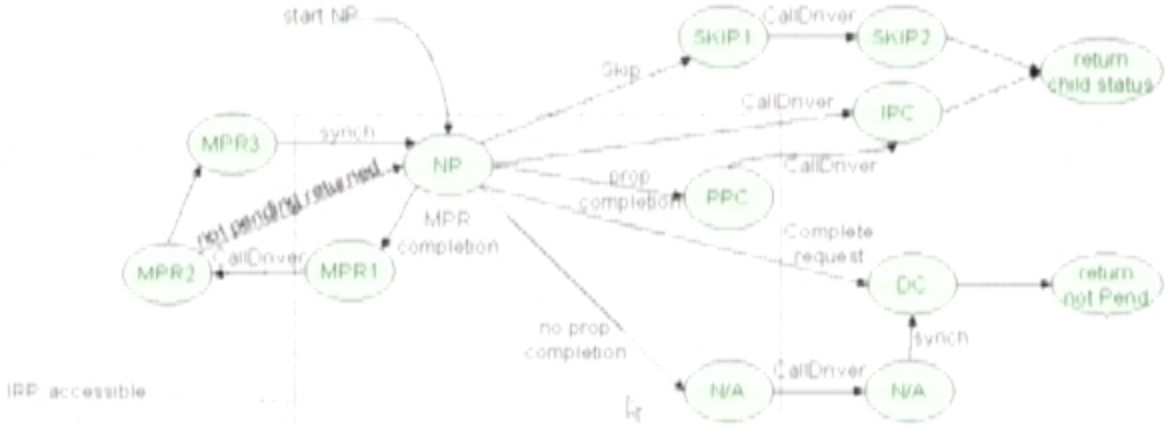


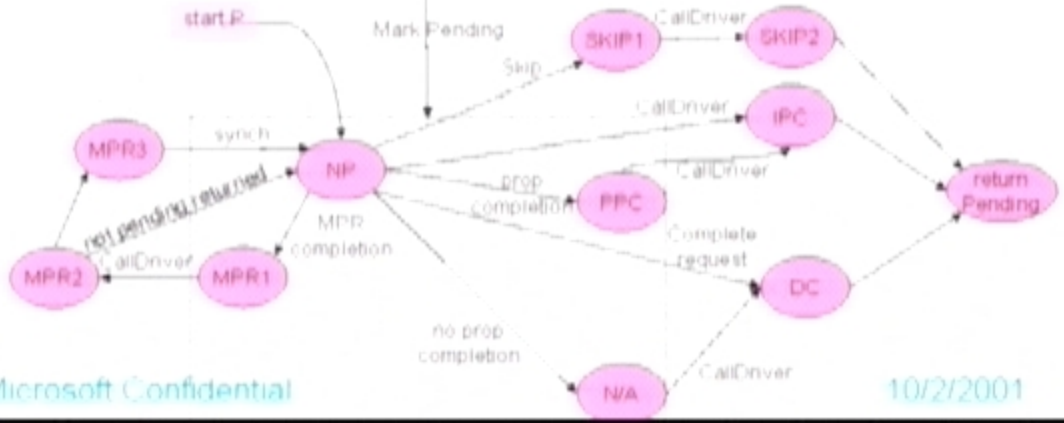
Static Driver Verifier

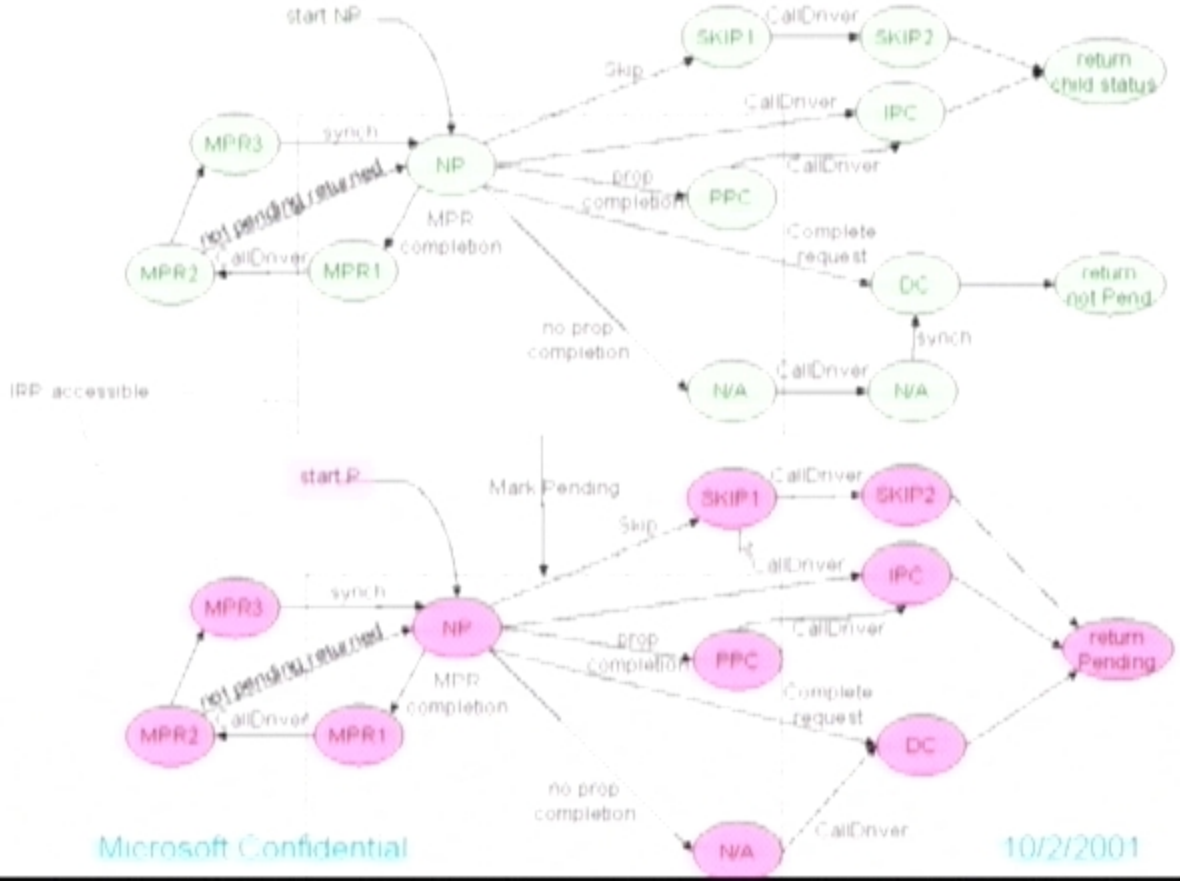


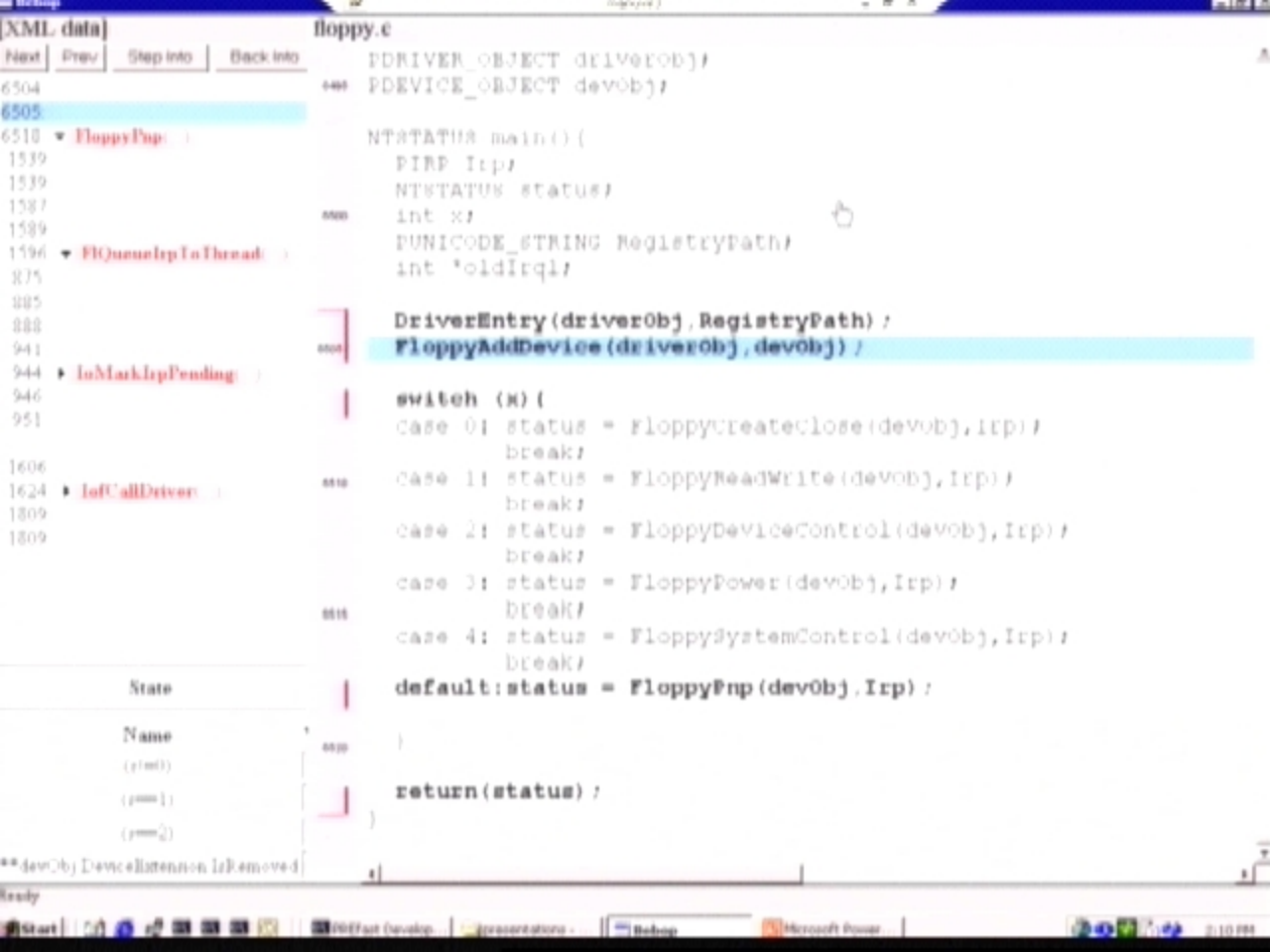
Static Driver Verifier (ii)

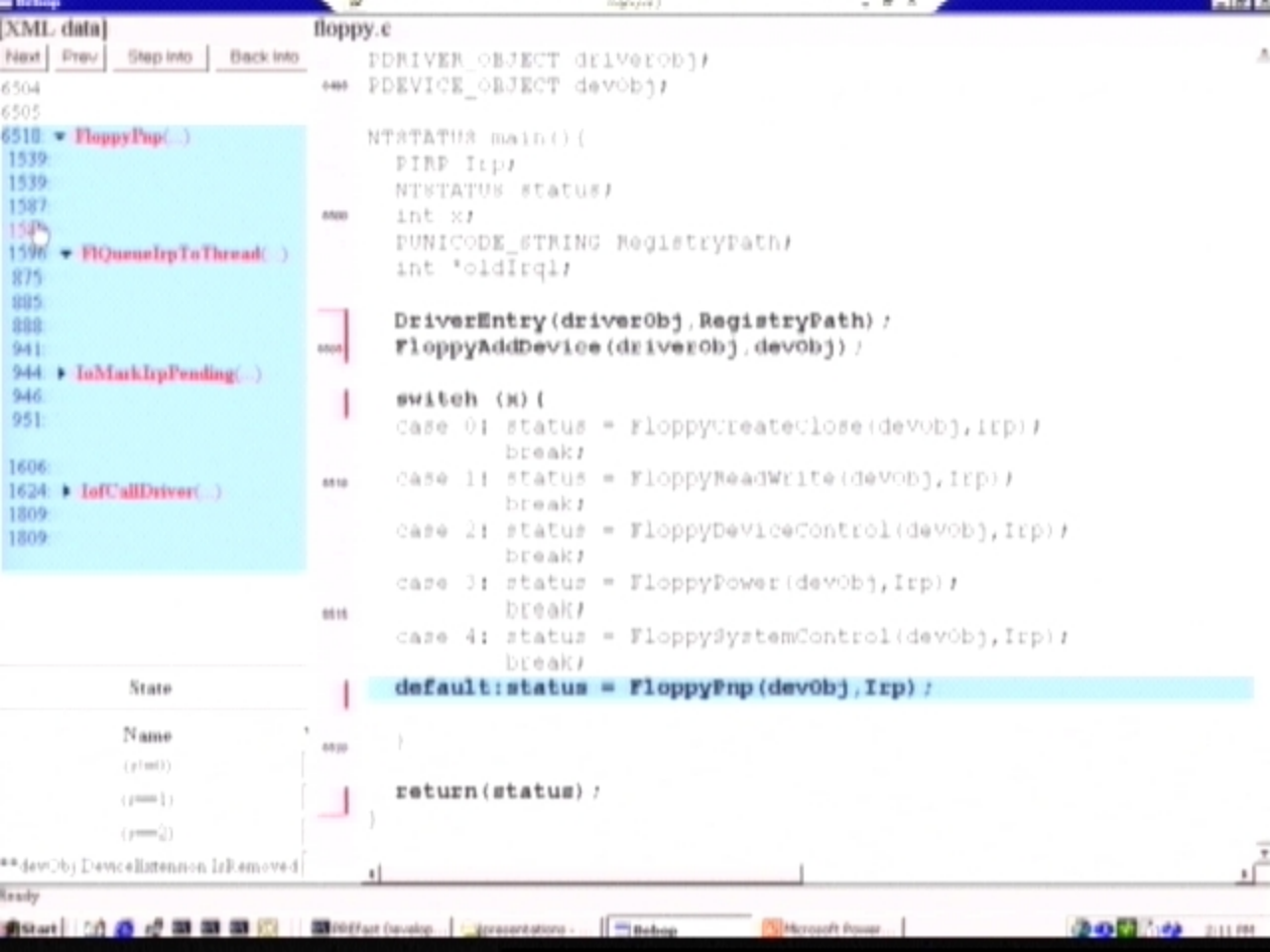


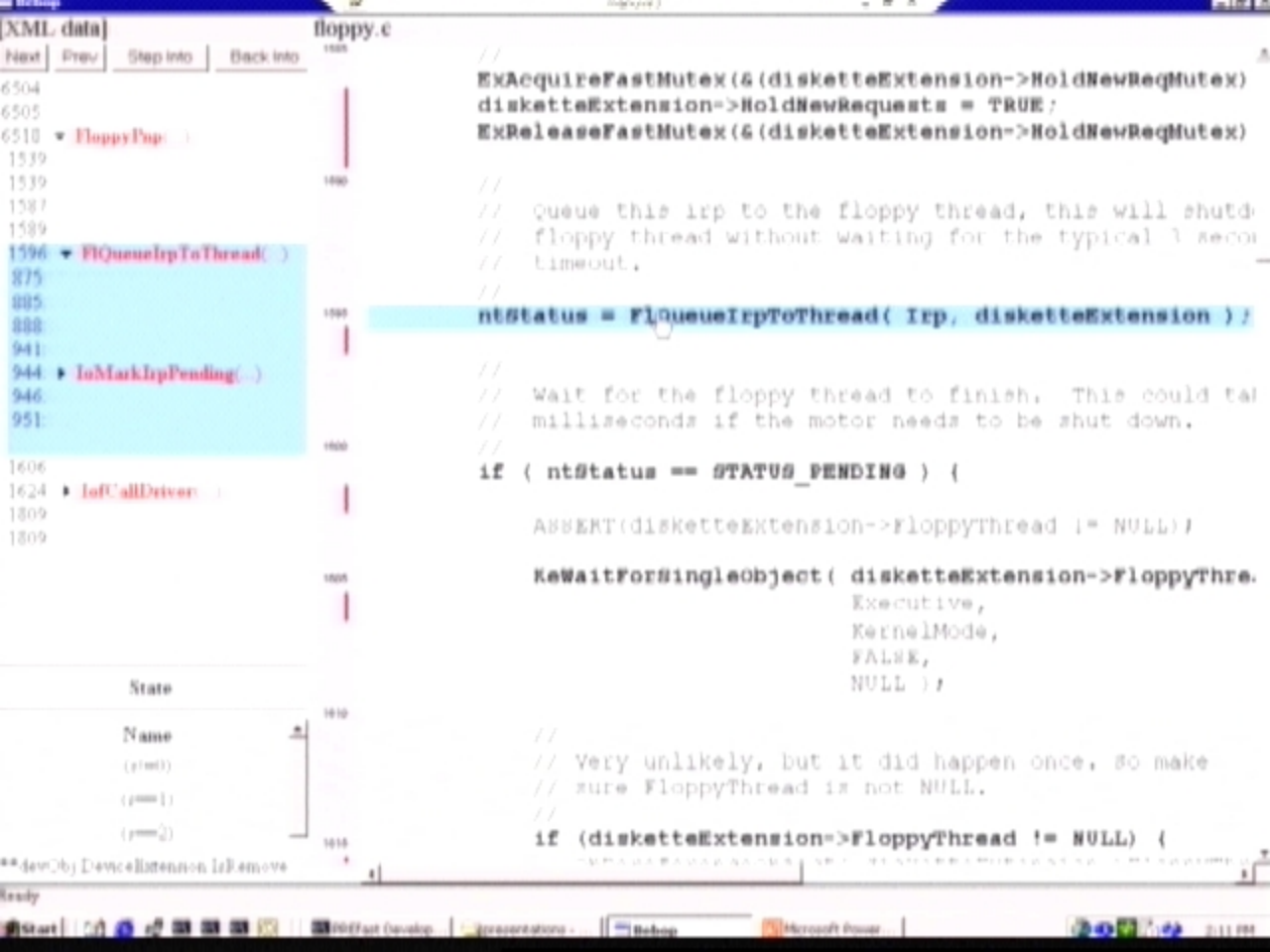

start NP
SKIP1
CallDriver
SKIP2
return child status
Skip
CallDriver
IPC
MPR3
synch
NP
CallDriver
prop completion
PPC
not pending / status read
MPR completion
Complete request
MPR2
CallDriver
MPR1
DC
return not Pend
synch
no prop completion
IRP accessible
N/A
CallDriver
N/A


start IP
Mark Pending
SKIP1
CallDriver
SKIP2
Skip
CallDriver
IPC
MPR3
synch
NP
CallDriver
prop completion
PPC
return Pending
not pending / status read
MPR completion
Complete request
MPR2
CallDriver
MPR1
DC
no prop completion
N/A
CallDriver


start NP
SKIP1
CallDriver
SKIP2
return child status
Skip
CallDriver
IPC
MPR3
synch
NP
CallDriver
prop completion
PPC
not pending/ status read
MPR completion
Complete request
MPR2
CallDriver
MPR1
DC
return not Pend
synch
no prop completion
IRP accessible
N/A
CallDriver
N/A
start IP
Mark Pending
SKIP1
CallDriver
SKIP2
Skip
CallDriver
IPC
MPR3
synch
NP
CallDriver
return Pending
prop completion
PPC
not pending/ status read
MPR completion
Complete request
MPR2
CallDriver
MPR1
DC
no prop completion
N/A
CallDriver


[XML data]
Next
Prev
Step Into
Back Into
floppy.c
PDRIVER_OBJECT driverObj;
PDEVICE_OBJECT devObj;
NTSTATUS main(){
PIRP Irp;
NTSTATUS status;
int x;
PUNICODE_STRING RegistryPath;
int *oldIrql;
DriverEntry(driverObj,RegistryPath);
FloppyAddDevice(driverObj,devObj);
switch (x){
case 0: status = FloppyCreateClose(devObj,Irp);
break;
case 1: status = FloppyReadWrite(devObj,Irp);
break;
case 2: status = FloppyDeviceControl(devObj,Irp);
break;
case 3: status = FloppyPower(devObj,Irp);
break;
case 4: status = FloppySystemControl(devObj,Irp);
break;
default:status = FloppyPnp(devObj,Irp);
}
return(status);
}
FloppyPnp
IoQueueIrpToThread
IoMarkIrpPending
IofCallDriver
State
Name
**devObj DeviceExtension IsRemoved
Ready


[XML data]
floppy.c
Next
Prev
Step into
Back into
6504
6505
6518 FloppyPnp(...)
1539
1539
1587
1596 FlQueueIrpToThread(...)
875
885
888
941
944 IoMarkIrpPending(...)
946
951
1606
1624 IofCallDriver(...)
1809
1809
State
Name
(y==0)
(y==1)
(y==2)
**devObj DeviceExtension IsRemoved
PDRIVER_OBJECT driverObj;
PDEVICE_OBJECT devObj;
NTSTATUS main(){
PIRP Irp;
NTSTATUS status;
int x;
PUNICODE_STRING RegistryPath;
int *oldIrql;
DriverEntry(driverObj,RegistryPath);
FloppyAddDevice(driverObj,devObj);
switch (x){
case 0: status = FloppyCreateClose(devObj,Irp);
break;
case 1: status = FloppyReadWrite(devObj,Irp);
break;
case 2: status = FloppyDeviceControl(devObj,Irp);
break;
case 3: status = FloppyPower(devObj,Irp);
break;
case 4: status = FloppySystemControl(devObj,Irp);
break;
default:status = FloppyPnp(devObj,Irp);
}
return(status);
}
Ready


[XML data]
floppy.c
Next
Prev
Step Into
Back Into
FloppyPnp( )
FlQueueIrpToThread( )
IoMarkIrpPending( )
IofCallDriver( )
State
Name
ExAcquireFastMutex(&(disketteExtension->HoldNewReqMutex)
disketteExtension->HoldNewRequests = TRUE;
ExReleaseFastMutex(&(disketteExtension->HoldNewReqMutex)
Queue this irp to the floppy thread, this will shutd
floppy thread without waiting for the typical 1 secon
timeout.
ntStatus = FlQueueIrpToThread(Irp, disketteExtension);
Wait for the floppy thread to finish. This could tal
milliseconds if the motor needs to be shut down.
if (ntStatus == STATUS_PENDING) {
ASSERT(disketteExtension->FloppyThread != NULL);
KeWaitForSingleObject(disketteExtension->FloppyThre
Executive,
KernelMode,
FALSE,
NULL);
Very unlikely, but it did happen once, so make
sure FloppyThread is not NULL.
if (disketteExtension->FloppyThread != NULL) {
Ready

[XML data]

Next | Prev | Step Into | Back Into

6504
6505
6510 ▼ FloppyPnp
1539
1539
1587
1589
1596 ▼ FlQueueIrpToThread
875
885
888
941
944 ▶ IoMarkIrpPending
946
951
1606
1624 ▶ IofCallDriver
1809
1809

State

Name

(=0)

(==1)

(==2)

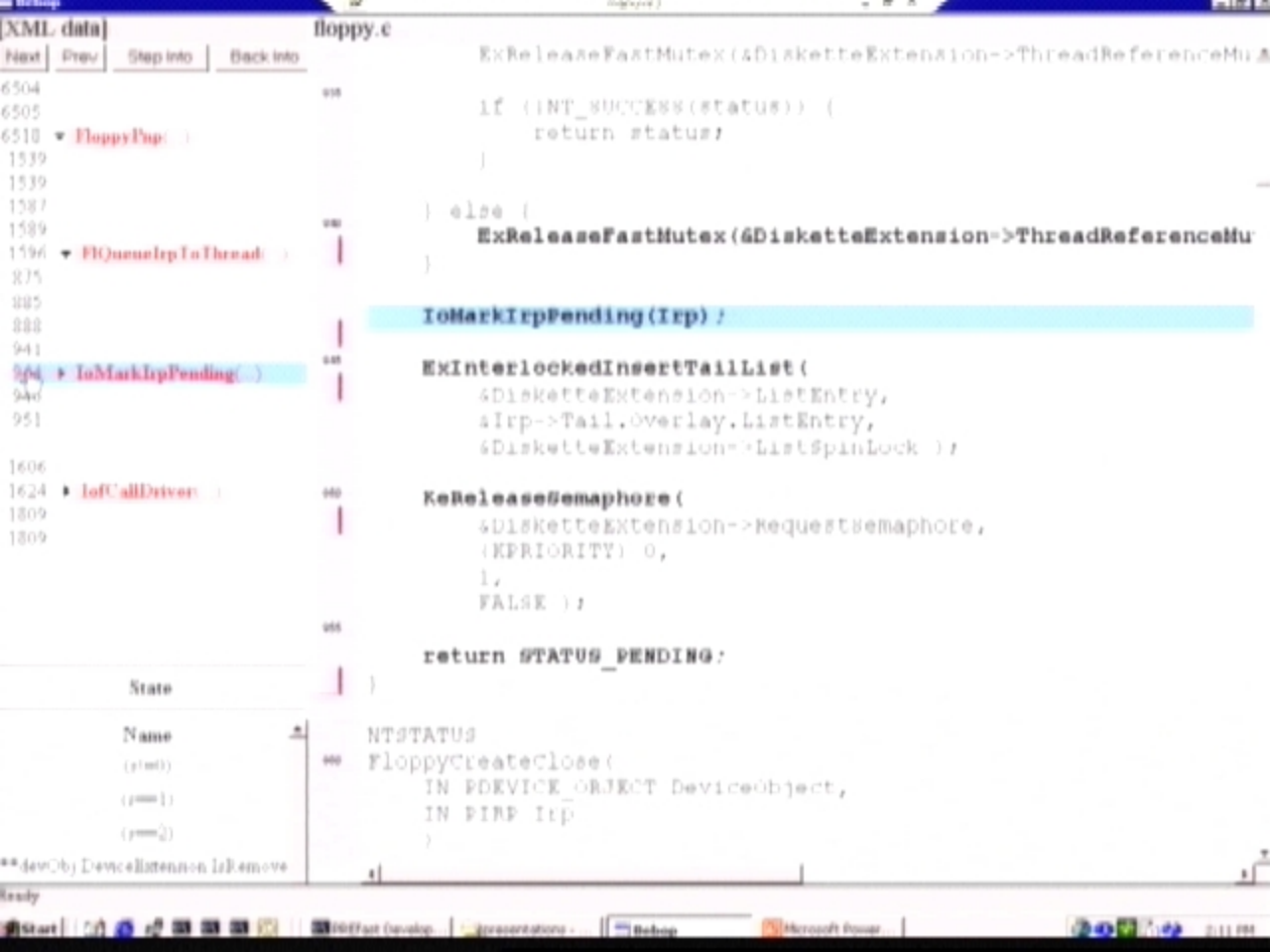
**devObj DeviceExtension IoRemove

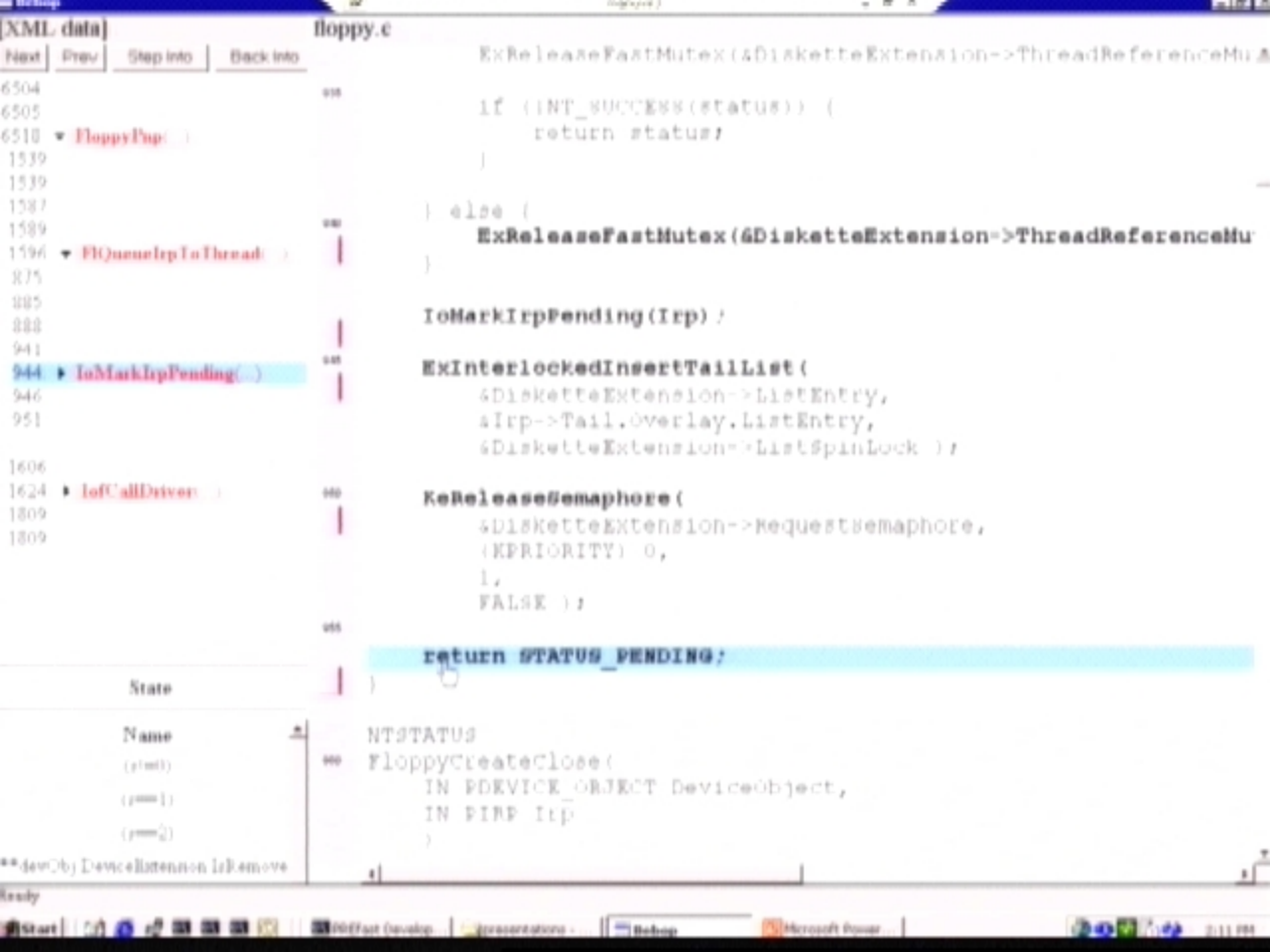
floppy.c

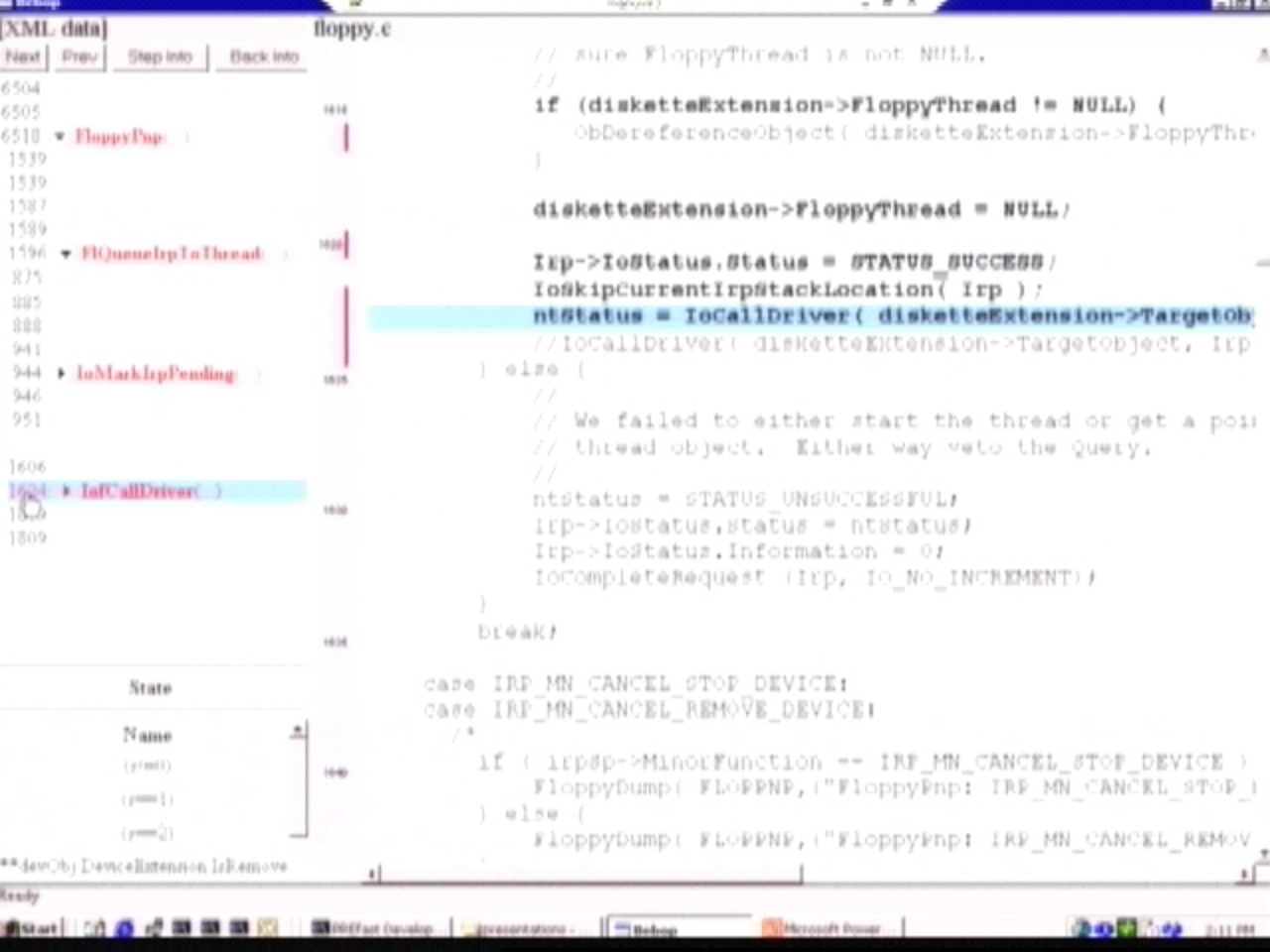
```
    KIRQL       oldIrql;
    NTSTATUS    status;
    HANDLE      threadHandle;
    PIO_STACK_LOCATION irpSp = IoGetCurrentIrpStackLocation( Irp

    //
    // Verify if the system is powering down. If so we fail
    // the irps.
    //
    ExAcquireFastMutex(&DisketteExtension->PowerDownMutex);
    if (DisketteExtension->PoweringDown == TRUE) {
        ExReleaseFastMutex(&DisketteExtension->PowerDownMutex);
        FloppyDump( FLOPDBGP,
                    ("Queue IRP: Bailing out since power irp is wa

        Irp->IoStatus.Status = STATUS_POWER_STATE_INVALID;
        Irp->IoStatus.Information = 0;
        return STATUS_POWER_STATE_INVALID;
    }
    ExReleaseFastMutex(&DisketteExtension->PowerDownMutex);
    FloppyDump( FLOPSHOW, ("Queue IRP: No power irp waiting.\n")

    ExAcquireFastMutex(&DisketteExtension->ThreadReferenceMutex)

    if (++(DisketteExtension->ThreadReferenceCount) == 0) {
        OBJECT_ATTRIBUTES ObjAttributes;

        DisketteExtension->ThreadReferenceCount++;

        FloppyResetDriverPaging();
```

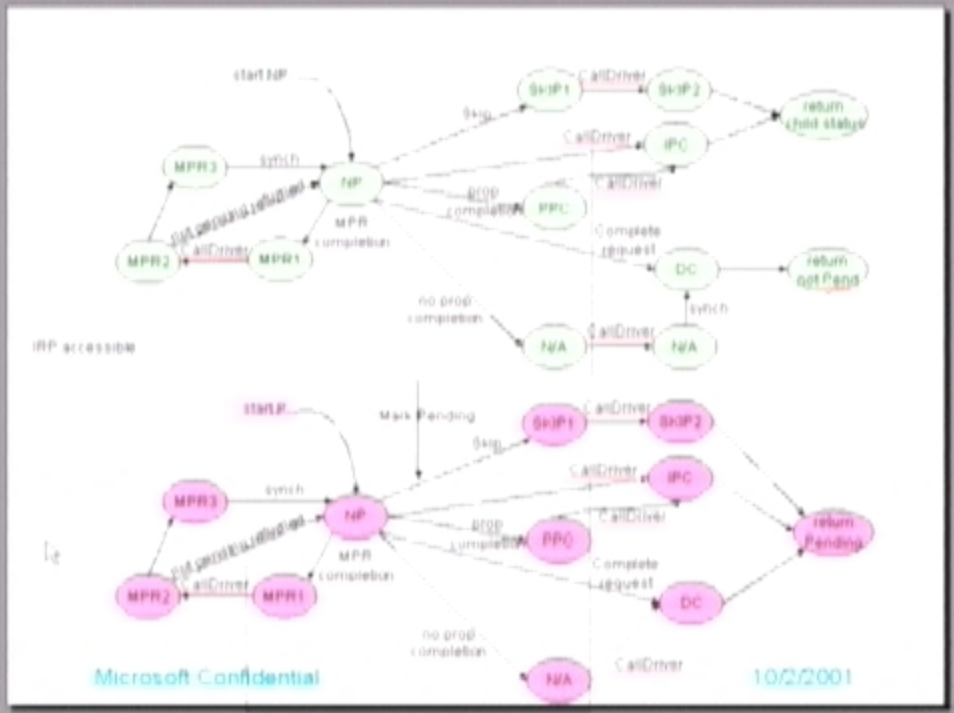
Ready

Start | Debug | Microsoft Power


[XML data]
floppy.c
Next
Prev
Step Into
Back Into
6504
6505
6510 FloppyPnp( )
1539
1539
1587
1589
1596 FlQueueIrpToThread( )
875
885
888
941
964 IoMarkIrpPending( )
940
951
1606
1624 IofCallDriver( )
1809
1809
State
Name
(==0)
(==1)
(==2)
**devObj DeviceExtension IsRemove
ExReleaseFastMutex(&Diskette Extension->ThreadReferenceMu
if (!NT_SUCCESS(status)) {
return status;
}
} else {
ExReleaseFastMutex(&Diskette Extension->ThreadReferenceMu
}
IoMarkIrpPending(Irp);
ExInterlockedInsertTailList(
&Diskette Extension->ListEntry,
&Irp->Tail.Overlay.ListEntry,
&Diskette Extension->ListSpinLock);
KeReleaseSemaphore(
&Diskette Extension->RequestSemaphore,
(KPRIORITY) 0,
1,
FALSE);
return STATUS_PENDING;
}
NTSTATUS
FloppyCreateClose(
IN PDEVICE_OBJECT DeviceObject,
IN PIRP Irp
)
Ready
Start
Debug
Microsoft Power


[XML data]
floppy.c
Next
Prev
Step into
Back into
FloppyPnp(
FlQueueIrpToThread(
IoMarkIrpPending(
IofCallDriver(
State
Name
ExReleaseFastMutex(&Diskette Extension->ThreadReferenceMu
if (!NT_SUCCESS(status)) {
return status;
}
} else {
ExReleaseFastMutex(&DiskketteExtension->ThreadReferenceMu
}
IoMarkIrpPending(Irp);
ExInterlockedInsertTailList(
&DiskketteExtension->ListEntry,
&Irp->Tail.Overlay.ListEntry,
&DiskketteExtension->ListSpinLock);
KeReleaseSemaphore(
&DiskketteExtension->RequestSemaphore,
(KPRIORITY) 0,
1,
FALSE);
return STATUS_PENDING;
}
NTSTATUS
FloppyCreateClose(
IN PDEVICE_OBJECT DeviceObject,
IN PIRP Irp
)
Ready


[XML data]
floppy.c
Next
Prev
Step Into
Back Into
FloppyPnp
FlQueueIrpToThread
IoMarkIrpPending
IofCallDriver
State
Name
// Sure FloppyThread is not NULL.
if (disketteExtension->FloppyThread != NULL) {
ObDereferenceObject(disketteExtension->FloppyThr
disketteExtension->FloppyThread = NULL;
Irp->IoStatus.Status = STATUS_SUCCESS;
IoSkipCurrentIrpStackLocation(Irp);
ntStatus = IoCallDriver(disketteExtension->TargetOb
//IoCallDriver(disketteExtension->TargetObject, Irp
} else {
// We failed to either start the thread or get a poi
// thread object. Either way veto the Query.
ntStatus = STATUS_UNSUCCESSFUL;
Irp->IoStatus.Status = ntStatus;
Irp->IoStatus.Information = 0;
IoCompleteRequest (Irp, IO_NO_INCREMENT);
break;
case IRP_MN_CANCEL_STOP_DEVICE:
case IRP_MN_CANCEL_REMOVE_DEVICE:
/*
if (irpSp->MinorFunction == IRP_MN_CANCEL_STOP_DEVICE)
FloppyDump(FLOPPNP,("FloppyPnp: IRP_MN_CANCEL_STOP_
} else {
FloppyDump(FLOPPNP,("FloppyPnp: IRP_MN_CANCEL_REMOV
**devObj DeviceExtension IsRemove
Ready


start IRP
SKIP1
CallDriver
SKIP2
return child status
Skip
CallDriver
IPC
MPR3
synch
NP
CallDriver
Prop completion
PPC
MPR completion
Complete request
MPR2
CallDriver
MPR1
DC
return not Pend
no prop completion
synch
IRP accessible
N/A
CallDriver
N/A
start IP
Mark Pending
SKIP1
CallDriver
SKIP2
Skip
CallDriver
IPC
MPR3
synch
NP
CallDriver
Prop completion
PPC
return Pending
MPR completion
Complete request
MPR2
CallDriver
MPR1
DC
no prop completion
N/A
CallDriver
Microsoft Confidential
10/2/2001

[XML data]

Next | Prev | Step Into | Back Into

6504
6505
6518 ▾ FloppyPnp(...)
1539
1539
1587
1589
1596 ▾ FlQueueIrpToThread(...)
875
885
888
941
944 ▸ IoMarkIrpPending(...)
946
951

1606
1624 ▸ IofCallDriver(...)
1809
1809

State

Name

(pwd)

(pwd)

**devObj DeviceExtension IsRemoved

floppy.c

```
    PDRIVER_OBJECT driverObj;
0480 PDEVICE_OBJECT devObj;

    NTSTATUS main(){
      PIRP Irp;
      NTSTATUS status;
0500  int x;
      PUNICODE_STRING RegistryPath;
      int *oldIrql;

      DriverEntry(driverObj,RegistryPath);
0505  FloppyAddDevice(driverObj,devObj);

      switch (x){
      case 0: status = FloppyCreateClose(devObj,Irp);
              break;
0510  case 1: status = FloppyReadWrite(devObj,Irp);
              break;
      case 2: status = FloppyDeviceControl(devObj,Irp);
              break;
      case 3: status = FloppyPower(devObj,Irp);
              break;
0515  case 4: status = FloppySystemControl(devObj,Irp);
              break;
      default:status = FloppyPnp(devObj,Irp);

0520  }

      return(status);
    }
```

Ready

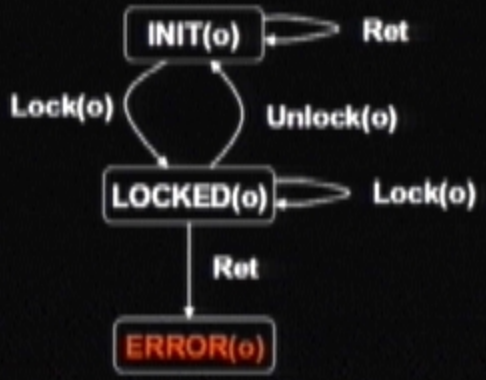
Start | 0001ast (se-ship) | presentations | Rebop | Microsoft Power

ESP (Manuvir Das)

- Error detection via scalable program analysis
- Similar to SLAM:
  - SLIC-like specification
  - Automatic analysis
  - Sound (finds all errors)
- Differences from SLAM:
  - Targeted at large (million+ LOC) C++ code bases
  - Only parameterized finite state protocols
  - Weaker (less precise) analysis

Parameterized Protocols

- FSM attached to data items
- Syntactic patterns trigger transitions



C++ code pattern	Transition
o->m_spinlock.Get()	Lock(o)
o->m_spinlock.Release()	Unlock(o)
return o	Ret

ESP Insight

- Three distinct entities
  - Sequence of actions along control flow path
  - Data involved in actions
    - o->Get(); p = o; p->Release();
  - Data involved in path feasibility
    - if (c) o->Get(); .... if (c) o->Release();
- Insight: use different static analyses to track each entity

Three Phases of Static Analysis

1. Global value flow analysis
   - Use GOLF to build call and value flow graph
2. Global protocol tracking
   - Use dataflow and value flow to track protocols associated with data items
3. Local feasibility analysis
   - Use local abstract simulation + summaries from Phase II to rule out infeasible paths

Example From SQL

```
void LockObject::Foo() {
    m_spinlock.Get();

    if (cond) {
        upwakeup0(&m_spinlock);
        return;
    }

    m_spinlock.Release();
}
void upwakeup0(Lock *lock) {
     lock->Release();
}
```

Example From SQL

```
void LockObject::Foo() {
    m_spinlock.Get();

    if (cond) {
        upwakeup0(&m_spinlock);
        return;
    }

    m_spinlock.Release();
}
void upwakeup0(Lock *lock) {
    lock->Release();
}
```

Phase 1
- Release called on pointer target of first parameter

Phase 2
- dataflow summary of FSM behavior
  *par1 : Locked -> Init, Init -> Error

Example From SQL

```
void LockObject::Foo() {
    m_spinlock.Get();

    if (cond) {
        upwakeup0(&m_spinlock);
        return;
    }

    m_spinlock.Release();
}
void upwakeup0(Lock *lock) {
    lock->Release();
}
```

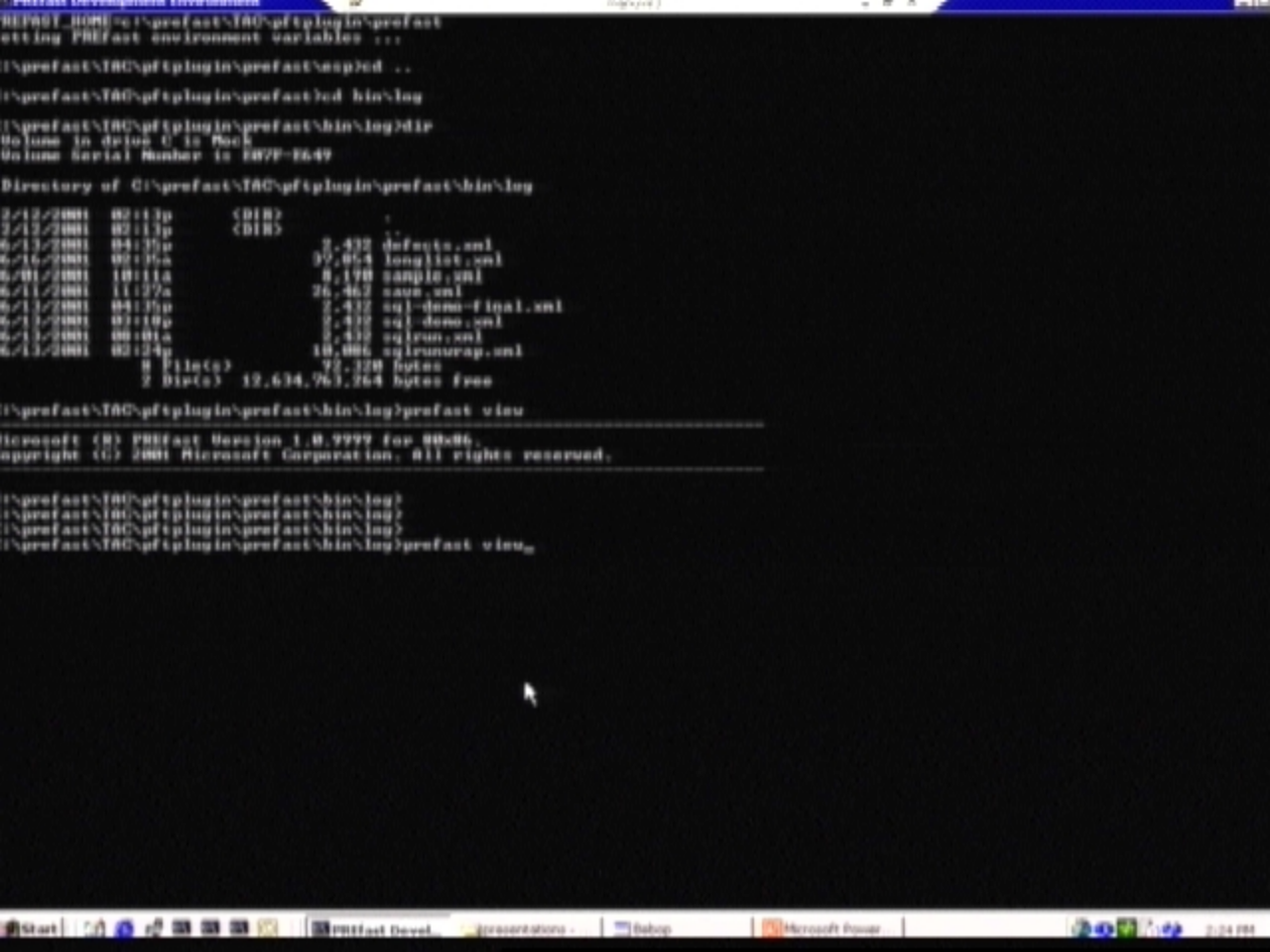
Phase 1
- *arg1 of upwakeup0 is m_spinlock
- Get and Release called on m_spinlock

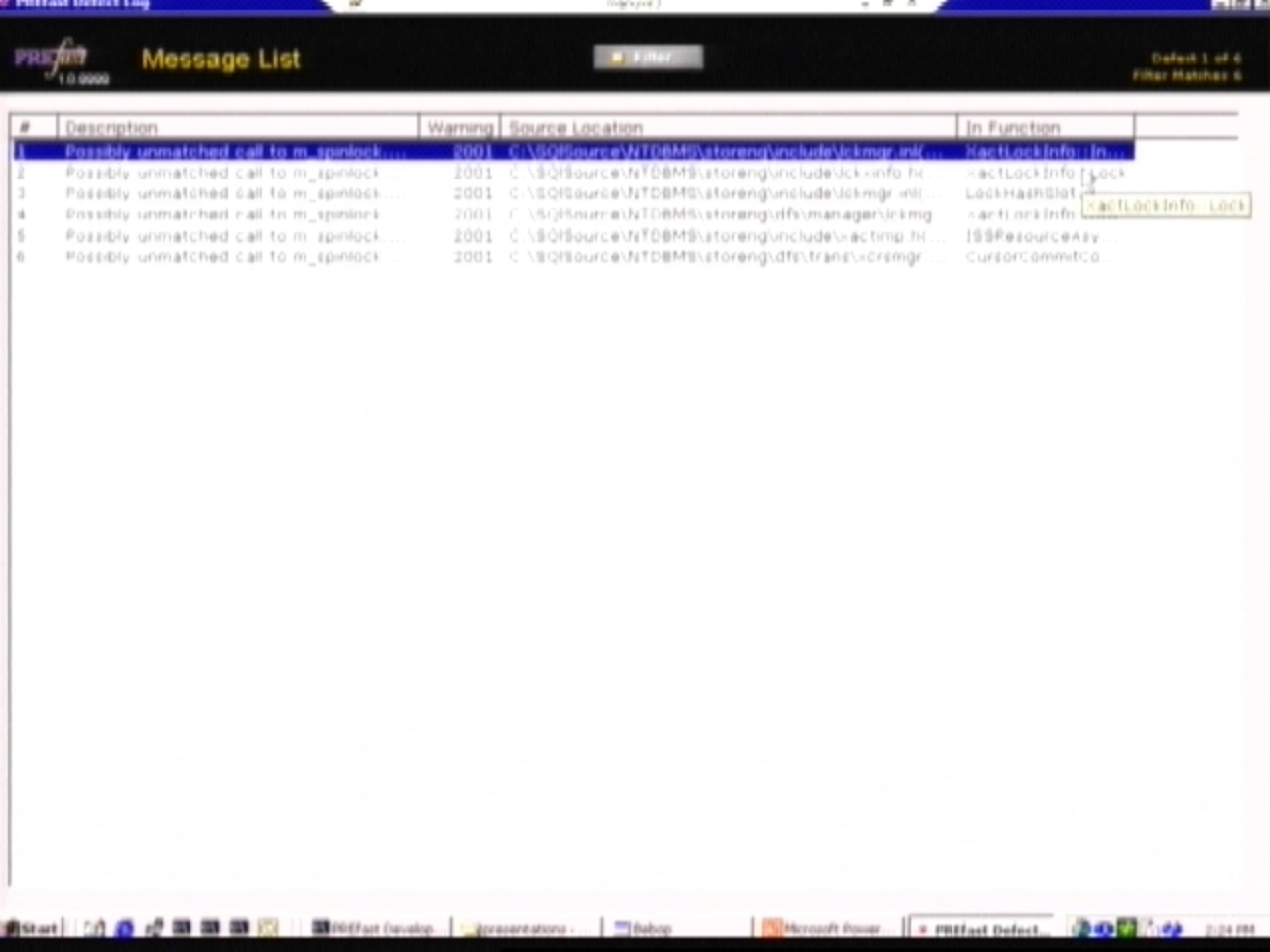
Phase 1
- Release called on pointer target of first parameter

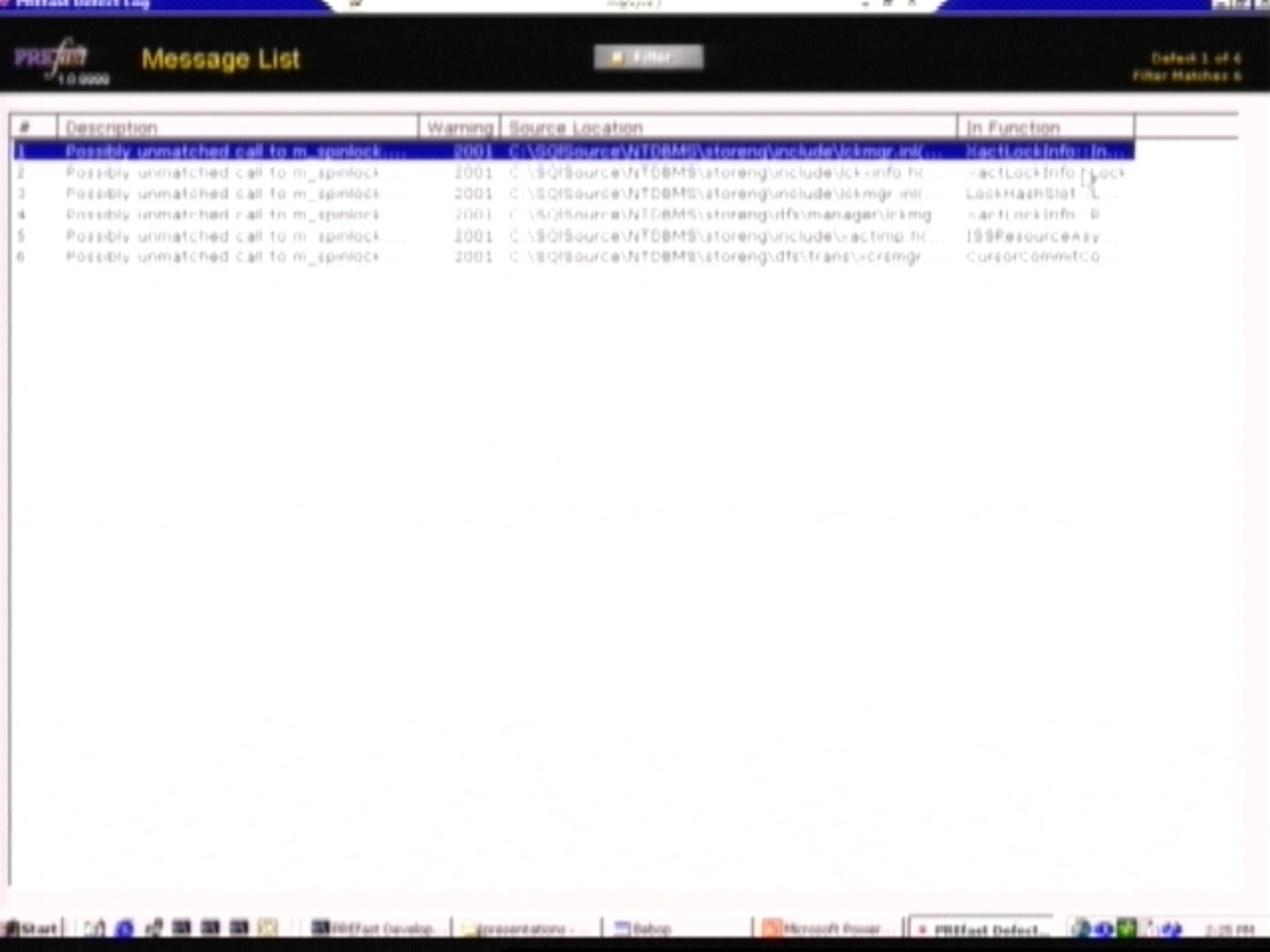
Phase 2
- dataflow summary of FSM behavior
  *par1 : Locked -> Init, Init -> Error

ESP Summary

- Early implementation for C
  - Verified stdio usage (15 files) in 150K LOC program
- Extend to full C++ and improve speed
- Hypothesis
  - For many (protocols) errors, scalable analysis is sufficiently precise
  - Intricate usage patterns are difficult for programmers, as well as tools


PREfast Developers Environment
PREFAST_HOME=c:\prefast\TAC\pftplugin\prefast
Setting PREfast environment variables ...
c:\prefast\TAC\pftplugin\prefast\esp>cd ..
c:\prefast\TAC\pftplugin\prefast>cd bin\log
c:\prefast\TAC\pftplugin\prefast\bin\log>dir
Volume in drive C is Mock
Volume Serial Number is E07F-E649
Directory of C:\prefast\TAC\pftplugin\prefast\bin\log
2/12/2001 02:13p <DIR> .
2/12/2001 02:13p <DIR> ..
6/13/2001 04:35p 2,432 defects.xml
6/16/2001 02:05a 37,854 longlist.xml
6/01/2001 10:11a 8,178 sample.xml
6/11/2001 11:37a 26,462 save.xml
6/13/2001 04:35p 2,432 sql-demo-final.xml
6/13/2001 03:10p 2,432 sql-demo.xml
6/13/2001 08:01a 2,432 sqlrun.xml
6/13/2001 02:24p 10,086 sqlrunwrap.xml
8 File(s) 92,320 bytes
2 Dir(s) 12,634,763,264 bytes free
c:\prefast\TAC\pftplugin\prefast\bin\log>prefast view
Microsoft (R) PREfast Version 1.0.9999 for 80x86
Copyright (C) 2001 Microsoft Corporation. All rights reserved.
c:\prefast\TAC\pftplugin\prefast\bin\log>
c:\prefast\TAC\pftplugin\prefast\bin\log>
c:\prefast\TAC\pftplugin\prefast\bin\log>
c:\prefast\TAC\pftplugin\prefast\bin\log>prefast view
Start
PREfast Devel...
presentations - ...
Bishop
Microsoft Power...


Message List
Filter
Defect 1 of 6
Filter Matches 6
Description Warning Source Location In Function
1 Possibly unmatched call to m_spinlock.... 2001 C:\SQlSource\NTDBMS\storeng\include\lckmgr.inl(... xactLockInfo::In...
2 Possibly unmatched call to m_spinlock.... 2001 C:\SQlSource\NTDBMS\storeng\include\lck-info.h(... xactLockInfo::Lock
3 Possibly unmatched call to m_spinlock.... 2001 C:\SQlSource\NTDBMS\storeng\include\lckmgr.inl(... LockHashSlot...
4 Possibly unmatched call to m_spinlock.... 2001 C:\SQlSource\NTDBMS\storeng\dfs\manager\lckmg... xactLockInfo...
5 Possibly unmatched call to m_spinlock.... 2001 C:\SQlSource\NTDBMS\storeng\include\xactimp.h(... ISSResourceAsy...
6 Possibly unmatched call to m_spinlock.... 2001 C:\SQlSource\NTDBMS\storeng\dfs\trans\xcremgr... CursorCommitCo...
xactLockInfo::Lock
Start
Microsoft Power...
PREfast Defect...


Message List
Filter
Defect 1 of 6
Filter Matches 6
Description Warning Source Location In Function
1 Possibly unmatched call to m_spinlock.... 2001 C:\SQlSource\NTDBMS\storeng\include\lckmgr.inl(... xactLockInfo::In...
2 Possibly unmatched call to m_spinlock.... 2001 C:\SQlSource\NTDBMS\storeng\include\lck-info.h(... xactLockInfo::Lock
3 Possibly unmatched call to m_spinlock.... 2001 C:\SQlSource\NTDBMS\storeng\include\lckmgr.inl(... LockHashSlot::L...
4 Possibly unmatched call to m_spinlock.... 2001 C:\SQlSource\NTDBMS\storeng\dfs\manager\lckmg... xactLockInfo::D...
5 Possibly unmatched call to m_spinlock.... 2001 C:\SQlSource\NTDBMS\storeng\include\xactmgr.h(... ISSResourceAsy...
6 Possibly unmatched call to m_spinlock.... 2001 C:\SQlSource\NTDBMS\storeng\dfs\trans\xcrmgr.... CursorCommitCo...

PREfast 1.0.0000

View Annotated Source

Prev | Msg List | Next

Defect 1 of 6
Filter Matches 6

View ...

Show Entire File

Go to ...

Start of Function

Start of Path

Warning Line

warning 2001 : Possibly unmatched call to m_spinlock.Get()

File path: c:\src\source\NTDBMS\storeng\include\lckmgr.inl
Function: XactLockInfo::InsertIntoLockQueue
Line: 1717

PREfast analysis path begins

```
1717            m_spinlock.Get ();
```

lckmgr.inl(1717) : warning 2001: Possibly unmatched call to m_spinlock.Get()
problem occurs in function 'XactLockInfo::InsertIntoLockQueue'

View ...

Show Entire File

Go to ...

Start of Function

Start of Path

Warning Line

PREfast analysis path begins

(1) m_spinlock.Get ();

lckmgr.inl(1717) : warning 28001: Possibly unmatched call to m_spinlock.Get()
problem occurs in function 'XactLockInfo::InsertIntoLockQueue'

PREfast 1.0.0000

View Annotated Source

Prev | Msg List | Next

Defect 1 of 4
Filter Matches 4

View ...

Show Entire File

Go to ...

Start of Function

Start of Path

Warning Line

PREfast analysis path begins

(1) m_spinlock.Get ();

lckmgr.inl(1717) : warning 28001: Possibly unmatched call to m_spinlock.Get() problem occurs in function 'XactLockInfo::InsertIntoLockQueue'

Start | PREfast (no-alop) | presentations | Bakop | Microsoft Power | PREfast Defect...

PREfast 1.0.0000 View Annotated Source

Prev | Msg List | Next

Defect 2 of 6
Filter Matches 6

View ...

Show Entire File

Go to ...

Start of Function

Start of Path

Warning Line

warning 2001 : Possibly unmatched call to m_spinlock.Get()
File path: c:\sql\source\NTDBMS\storeng\include\lck.info.h
Function: XactLockInfo::Lock
Line: 495

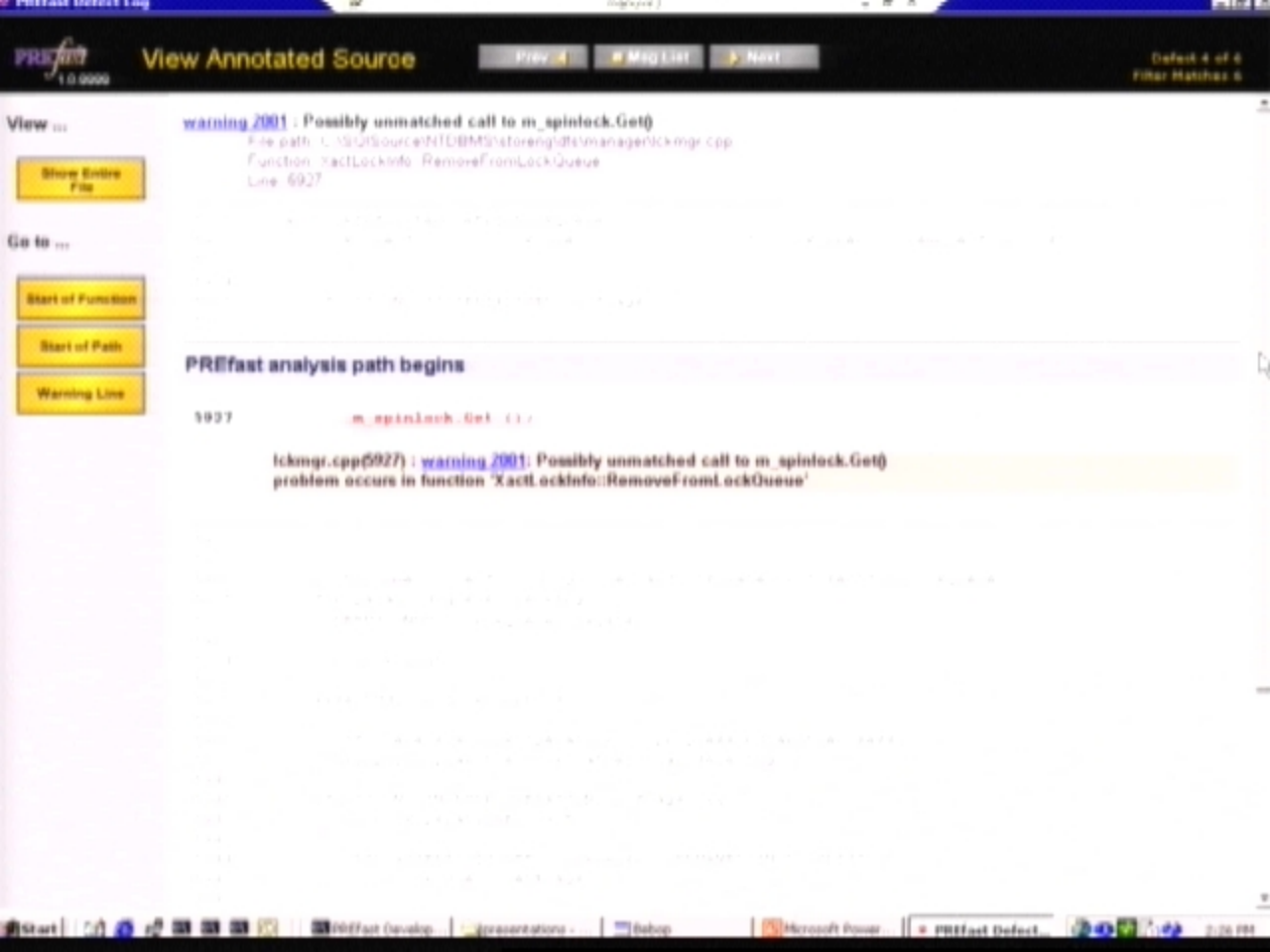
PREfast analysis path begins

495 m_spinlock.Get ();

lckxinfo.h(495) : warning 2001: Possibly unmatched call to m_spinlock.Get()
problem occurs in function 'XactLockInfo::Lock'

Start | PREfast (no-alop) | presentations | Inbox | Microsoft Power | PREfast Defect... | 2:26 PM


View Annotated Source
Prev
Msg List
Next
View Next Message in List
Defect 3 of 6
Filter Matches 6
View ...
Show Entire File
Go to ...
Start of Function
Start of Path
Warning Line
warning 2001 : Possibly unmatched call to m_spinlock.Get()
File path: L:\SQLSource\NTDBMS\storeng\include\lckmgr.inl
Function: LockHashSlot::LockSlot
Line: 279
PREfast analysis path begins
279 m_spinlock.Get ();
lckmgr.inl(279) : warning 2001: Possibly unmatched call to m_spinlock.Get()
problem occurs in function 'LockHashSlot::LockSlot'


View Annotated Source
Prev
Msg List
Next
View ...
Show Entire File
Go to ...
Start of Function
Start of Path
Warning Line
warning 2001 : Possibly unmatched call to m_spinlock.Get()
File path: L:\SQLSource\NTDBMS\storeng\dls\manager\lckmgr.cpp
Function: XactLockInfo::RemoveFromLockQueue
Line: 6927
PREfast analysis path begins
5927 m_spinlock.Get ();
lckmgr.cpp(5927) : warning 2001: Possibly unmatched call to m_spinlock.Get()
problem occurs in function 'XactLockInfo::RemoveFromLockQueue'

View ...

Show Entire File

Go to ...

Start of Function

Start of Path

Warning Line

File path: ...\SqlSource\NTDBMS\storeng\dfs\manager\lckmgr.cpp
Function: XactLockInfo::RemoveFromLockQueue
Line: 5927

```
(m_SERIALIZED, m_always)
```

PREfast analysis path begins

```
5927            m_spinlock.Get ();
```

lckmgr.cpp(5927) : warning 2001: Possibly unmatched call to m_spinlock.Get()
problem occurs in function 'XactLockInfo::RemoveFromLockQueue'

PREfast 1.0.0000

View Annotated Source

Prev 4 | Msg List | Next

Defect 4 of 6
Filter Matches 6

View ...

Show Entire File

Go to ...

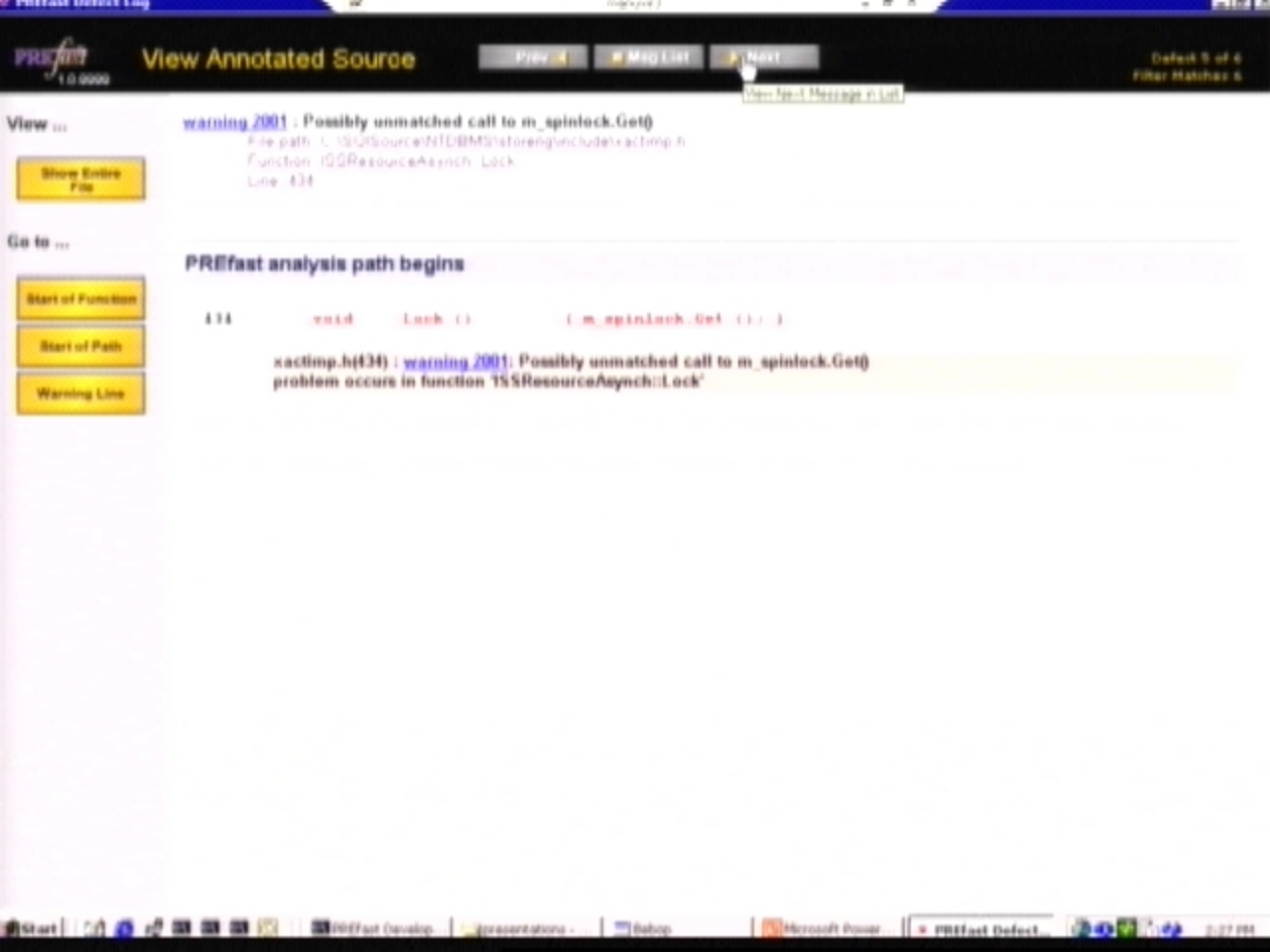
Start of Function

Start of Path

Warning Line

5927 m_spinlock.Get ();

lckmgr.cpp(5927) : warning 2001: Possibly unmatched call to m_spinlock.Get()
problem occurs in function 'XactLockInfo::RemoveFromLockQueue'


View Annotated Source
Prev
Msg List
Next
View Next Message in List
View ...
Show Entire File
Go to ...
Start of Function
Start of Path
Warning Line
warning 2001 : Possibly unmatched call to m_spinlock.Get()
File path: \SQLSource\NTDBMS\storeng\include\xactimp.h
Function: ISSResourceAsynch::Lock
Line: 434
PREfast analysis path begins
434 void Lock () { m_spinlock.Get (); }
xactimp.h(434) : warning 2001: Possibly unmatched call to m_spinlock.Get()
problem occurs in function 'ISSResourceAsynch::Lock'

View ...

Show Entire File

Go to ...

Start of Function

Start of Path

Warning Line

warning /001 : Possibly unmatched call to m_spinlock.Get()

File path: \\SQLSource\NTDBMS\storeng\dts\trans\xcrsmgr.cpp
Function: CursorCommitCoordinator::DoneEndTran
Line: 343

PREfast analysis path begins

```
343         m_spinlock.Get ();
```

xcrsmgr.cpp(343) : warning /001: Possibly unmatched call to m_spinlock.Get()
problem occurs in function 'CursorCommitCoordinator::DoneEndTran'

View ...

Show Entire File

Go to ...

Start of Function

Start of Path

Warning Line

warning /00l : Possibly unmatched call to m_spinlock.Get()

File path: C:\SQLSource\NTDBMS\storeng\dls\trans\xcrsmgr.cpp

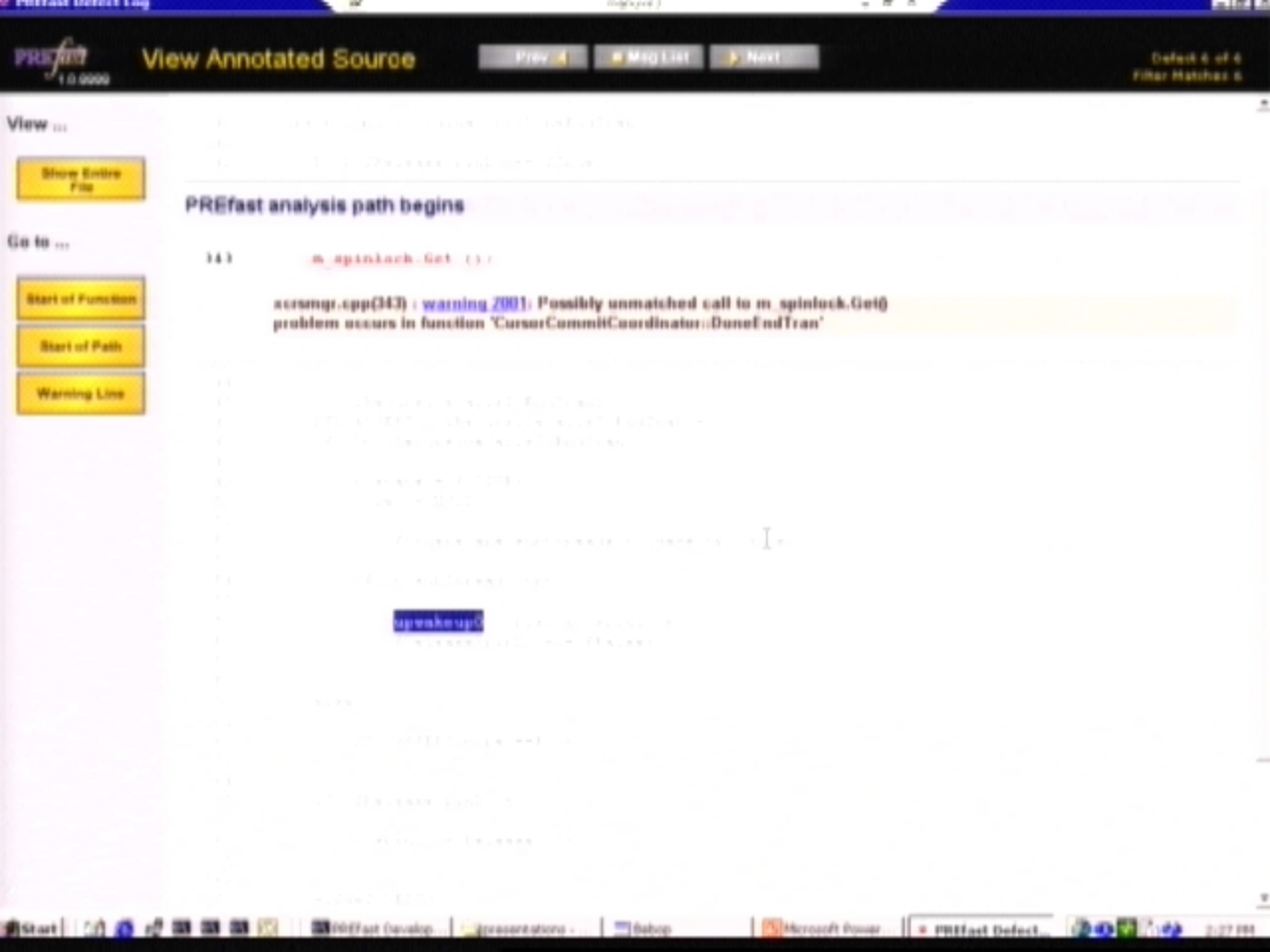
Function: CursorCommitCoordinator::DoneEndTran

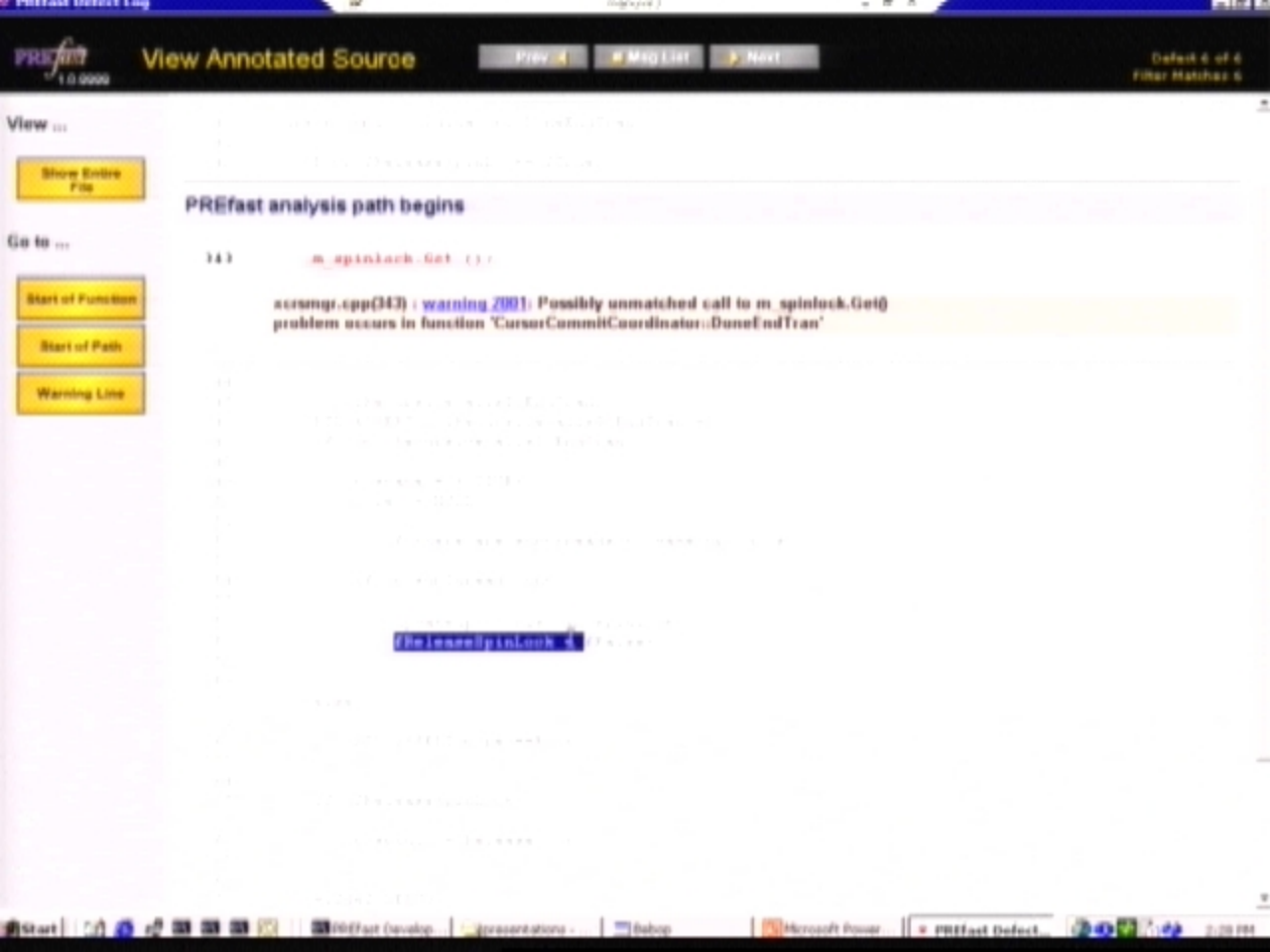
Line: 343

PREfast analysis path begins

```
343     m_spinlock.Get ();
```

xcrsmgr.cpp(343) : warning /00l: Possibly unmatched call to m_spinlock.Get()
problem occurs in function 'CursorCommitCoordinator::DoneEndTran'


View Annotated Source
Prev
Msg List
Next
View ...
Show Entire File
Go to ...
Start of Function
Start of Path
Warning Line
PREfast analysis path begins
343 m_spinlock.Get ();
accmgr.cpp(343) : warning 7001: Possibly unmatched call to m_spinlock.Get()
problem occurs in function 'CursorCommitCoordinator::DoneEndTran'


View Annotated Source
Prev
Msg List
Next
View ...
Show Entire File
Go to ...
Start of Function
Start of Path
Warning Line
PREfast analysis path begins
m_spinlock.Get ();
acrsmgr.cpp(343) : warning 28001: Possibly unmatched call to m_spinlock.Get()
problem occurs in function 'CursorCommitCoordinator::DoneEndTran'
ReleaseSpinLock ();

PREfast 10.0000

View Annotated Source

Prev | Msg List | Next

Defect 6 of 6
Filter Matches 6

View ...

Show Entire File

Go to ...

Start of Function

Start of Path

Warning Line

PREfast analysis path begins

143 m_spinlock.Get ();

xcrsmgr.cpp(343) : warning 28001: Possibly unmatched call to m_spinlock.Get()
problem occurs in function 'CursorCommitCoordinator::DoneEndTran'

Start | PREfast (w-alop... | presentations ... | Bakup | Microsoft Power... | PREfast Defect...

Concurrency

- Parallel and concurrent programs are hard to write, debug, test, modify, …
  - Everyone is going to be writing many more of them
  - Cheap SMPs and .net
- Asynchronous programming
  - Message-passing (non-RPC) communication
  - Hailstorm, Indigo, …
  - BizTalk
- Difficult behavioral properties
  - Deadlock-freedom
  - Communication progress
  - Message understood
- Few tools

Sharpie Example

```
// Declaration of client
operation client(In ch: z, Out ch: w) @ z? -> w!

// Implementation of server
server (In ch: x, Out ch: y) @ x? -> y!
{
  select x?() ->
    async c, d. client(c, d)
    in
       select d?() -> y!
}
```

Sharpie Example

```
// Declaration of client
operation client(In ch: z, Out ch: w) @ z? -> w!

// Implementation of server
server (In ch: x, Out ch: y) @ x? -> y!
{
  select x?() ->
    async c, d, client(c, d)
    in
       select d?() -> y!
}
```

Receive on z;
Send on w

Async call on client

Sharpie Example

```
// Declaration of client
operation client(In ch: z, Out ch: w) @ z? -> w!

// Implementation of server
server (In ch: x, Out ch: y) @ x? -> y!
{
  select x?() ->
    async c, d, client(c, d)
    in
       select d?() -> y!
}
```

Receive on z;
Send on w

Async call on client

Receive on d

Security Analysis

- Many security flaws exploit information flow

```
UserInput(s1);
...
CallToKernel(v);
```

- Value flow analysis
  - Determines where value comes from or goes to during execution
  - "Can the value of string s1 flow to variable v during execution?"
  - Previously, interprocedural analysis was imprecise or expensive

GOLF (Manuvir Das, Manuel Fahndrich, Jakob Rehof)

- Generalized One Level Flow
  - Whole-program value flow engine
- Scalable, context-sensitive, conservative analysis
  - Handles function calls accurately (context-sensitive)
  - Efficient (MLOC in seconds)
  - Precise (experimental evidence)
- Algorithms incorporated into PREfix
- Basis for family of security tools

SPT Summary

- SPT approach
  - Programmer/tester writes partial specifications
  - Tools ensure code follows specs
    - SLAM: extremely precise analysis
    - ESP: trade precision for scalability
    - Vault: integrate approach into language
    - Behavet: concurrency
    - GOLF: value flow problems
- Why so many projects?
  - Partial verification, not total correctness